On the optimal block size for block-based, motion-compensated video coders*

Jordi Ribas-Corbera

Sharp Labs of America, 5750 NW Pacific Rim Blvd,
Camas, WA 98607, USA. E-mail: jordi@sharplabs.com

David L. Neuhoff

EECS Department, University of Michigan,
Ann Arbor, MI 48109, USA. E-mail: neuhoff@eecs.umich.edu

ABSTRACT

In block-based video coding, the current frame to be encoded is decomposed into blocks of the same size, and a motion vector is used to improve the prediction for each block. The motion vectors and the difference frame, which contains the blocks' prediction errors, must be encoded with bits. Typically, choosing a smaller block size (using more motion vectors) will improve the prediction and hence decrease the number of difference frame bits, but it will increase the number of motion bits since more motion vectors need to be encoded. Not surprisingly, there must be some value for the block size that optimizes the tradeoff between motion and difference frame bits, and thus minimizes their sum (the total number of bits for the frame). Despite the widespread experience with block-based video coders, there is little analysis or theory that quantitatively explains the effect of block size on encoding bit rate, and ordinarily the block size for a coder is chosen based on empirical experiments on video sequences of interest. In this work, we derive a procedure to determine the optimal block size that minimizes the encoding rate for a typical block-based video coder. To do this, we analytically model the effect of block size and derive expressions for the encoding rates for both motion vectors and difference frames, as functions of block size. Minimizing these expressions leads to a simple formula that indicates how to choose the block size in these types of coders. This formula also shows that the best block size is a function of the accuracy with which the motion vectors are encoded and several parameters related to key characteristics of the video scene, such as image texture, motion activity, interframe noise, and coding distortion. We implement the video coder and use our analysis to optimize and explain its performance on real video frames.

Keywords: video coding, motion estimation, motion field resolution, block size, motion vector accuracy.

1 INTRODUCTION

In block-based, motion-compensated video coding, a motion field with one motion vector per image block is used to improve the prediction of the frame to be encoded. These motion vectors must themselves be encoded with bits. The benefit of this strategy is that the sum of motion bits plus those used to encode the motion-compensated difference frame is often significantly less than the number of bits needed to encode the difference frame without motion compensation.

*This work was supported by NSF Grant NCR-9415754.

The number of motion bits depends on the number of motion vectors and the accuracies with which they are encoded. Increasing either will improve the prediction and hence decrease the number of bits needed to encode the difference frame. Up to a point, it will also decrease the total number of bits (i.e., the motion bits plus the difference frame bits).

In this paper we focus on the classical coding scheme where in each frame all blocks have the same size of $B \times B$ pixels (e.g., 4×4, 8×8, 16×16) and all motion vectors are encoded with the same pixel accuracy Δ (e.g., $\Delta = 1$, $1/2$, $1/4$).[1–3,12,13,16,17] In these coders, a key issue is to find the block size B^* and the motion vector accuracy Δ^* that minimize the total encoding rate (in bits/pixel) for a given level of distortion.

Ordinarily, the choice of B and Δ is based on heuristics or empirical experiments on scenes of interest. For example, most block-based video coders use pixel accurate ($\Delta = 1$) or half-pixel accurate ($\Delta = 1/2$) motion vectors.[1–4,12,13] MPEG[2] and H.261[3] use block size $B = 16$. Recently, it was found that $B = 2$ was best for ultrasound video,[13] and that, with the appropriate motion coder, $B = 8$ was better than $B = 16$ for some typical video scenes.[12] In H.263,[4] there is the option of splitting 16×16 blocks into four 8×8 subblocks. But despite the widespread experience with block-based video coders, there has been little analysis or theory that predicts or explains how the best choices of B and Δ depend on the key characteristics of the video scene, such as image texture, motion activity, interframe noise, and coding distortion.

There has been a significant amount of work on finding methods to design object oriented or variable-block size motion fields for individual frames.[5–7,9–11] Though generally these methods could be applied to the fixed block size motion fields considered in this paper, they are fairly complex when based on rate-distortion optimizations[6,11] and suboptimal when based on simple ad hoc criteria or greedy optimizations.[5,7,9,10] Further, they do not provide analysis or theory that explains or predicts how the best block size depends on motion accuracy and the key scene characteristics.

The problem of finding the effect of the motion vector accuracy Δ on the difference frame was first considered by Girod.[8] Recent work of the present authors[16,17] focused on finding the optimal accuracy Δ^* for a fixed block size B in the context of certain simple, appropriate models. However, neither the functional dependence of Δ^* on B, nor the optimal block size B^*, were addressed.

In this paper, we extend our previous work[16,17] to find the optimal block size B^* as a function of Δ in the context of similar models. Also, we derive an expression for the optimal motion vector accuracy Δ^* as a function of B. Specifically, we consider a simple block-based video coder that uses scalar quantization with entropy coding of the difference frame and lossless differential encoding of the motion vectors. We find models for the rates of these coders as functions of block size B and motion accuracy Δ that are simple enough that we are able to derive formulas for the optimal B^* and Δ^*. These formulas show the effects of the key characteristics of the video scene on B^* and Δ^*. For example, they indicate that the block size should be smaller for higher textured frames and when the motion vectors are encoded with low accuracy. On the other hand, the block size should be larger for larger coding distortions and when the variance of the motion vectors is small or the motion vector correlation is large.

Finally, we implement the video coder and use these formulas to optimize and explain its performance on real video frames. The experimental results suggest that our equations are accurate and that significant bit rate savings can be achieved with our optimization procedures.

2 THE BLOCK-BASED VIDEO CODER

In a typical block-based video coder a prediction for the frame to be encoded is formed by subdividing the frame into blocks (e.g., 8×8 pixels per block), finding the *motion vector* for each block that displaces it to match a block in the previous frame[//], and then displacing the blocks in the decoded previous frame according to the motion vectors. Subtracting the predicted frame from the actual frame yields a *difference frame.*

In our coder, the difference frame pixels are encoded by a uniform scalar quantizer with quantization step size Q followed by a first-order entropy coder where the code used is adapted to that particular difference frame. Such a coder is adequate for this study because of the low correlation between pixel values in the difference frame.[8,7] However, as in our previous work,[16,17] it is anticipated that the approach followed here could be extended to more advanced coders and that the general nature of the conclusions found here remains relevant.

The motion vectors are computed with accuracy Δ and are differentially encoded (DPCM) with lossless entropy coding. This technique is popular in block-based video coders since it has been shown to perform well in practice.[1-4,14] It is important to realize that the effect of B on the coder performance depends on the choice of motion vector coder. For example, observe that a motion vector coder that encodes four motion vectors produced for $B = 8$ in a lossy fashion may be equivalent to a motion vector coder that losslessly encodes motion vectors produced for $B = 16$ (c.f. Sipitca and Madisetti[12]), and these two coders will clearly have different optimal block sizes. Because of this, the equations and results in this work are only valid for the typical differential encoding of motion vectors. Nevertheless, other types of motion coders could also be considered within our framework by simply modifying the motion rate model.

2.1 Rate for the difference frame R_D

The expected rate produced by our difference frame coder when encoding a particular difference frame is $R_D \approx H(Q)$, where $H(Q)$ is the empirical entropy of the Q-quantized difference frame pixel values. We assume that the difference frame pixels have a discrete Laplacian form,[7] with variance σ^2 and use the approximate formula[16,17]:

$$H(Q) \approx \begin{cases} \log_2 \sqrt{2}e\sigma - \log_2 Q, & \sigma^2 > \frac{Q^2}{2e} \quad \text{(low distortion)} \\ \frac{e}{Q^2 \ln 2}\,\sigma^2, & \sigma^2 \leq \frac{Q^2}{2e} \quad \text{(high distortion).} \end{cases} \tag{1}$$

Note that σ^2 will be larger than $Q^2/(2e)$ for small values of Q (low distortion), and smaller for large values of Q (high distortion). The approximation in (1) for low distortion is obtained by relating the entropy $H(Q)$ to the differential entropy of a Laplacian,[15] and for high distortion by choosing the linear function of σ^2 to meet the log function at a point of equal slope, which turns out to be at $\sigma^2 = Q^2/(2e)$. From now on, we focus on the more interesting high distortion case ($Q^2 \geq 2e\sigma^2$), which typically corresponds to PSNR's of 40 dB or less.

As our estimate of σ^2 we choose:

$$\frac{1}{P}\sum_{i=1}^{P/B^2} E_i, \tag{2}$$

where E_i is the energy of the ith block in the difference frame, and P and P/B^2 are the total number of pixels and blocks in a frame, respectively. It has recently been shown[16,17] that for pixel or subpixel motion vector accuracies, i.e., $\Delta \leq 1$, E_i can be approximated by the quadratic formula:

$$E_i \approx A_i\Delta^2 + C_i, \tag{3}$$

where

$$A_i = \frac{1}{12}\sum_{(x,y)\epsilon\text{Block } i} \left((F(x,y) - F(x+1,y))^2 + (F(x,y) - F(x,y+1))^2\right), \tag{4}$$

[//] Our analysis can also apply to backward or interpolative prediction.

with $F(x,y)$ the intensity value of the pixel at coordinates (x,y) in the current frame F. Note that A_i is a simple measure of the texture in the ith block, and C_i is the energy in the ith block of the difference frame if the block's motion vector was found with infinite accuracy ($\Delta = 0$). Clearly, C_i would be zero if an exact match of the ith block could be found in the previous frame, but in practice there is interframe noise and other phenomena and $C_i > 0$. Specifically, we model C_i as follows:

$$C_i = C_{i,0} + \frac{Q^2}{12}B^2 + S_i, \tag{5}$$

where $C_{i,0}$ is the interframe noise energy in the block (e.g., due to light changes, occlusions, camera noise, etc.), $Q^2/12$ is the approximate coding distortion introduced in the encoded previous frame, and S_i is the error energy caused by having several pixel motions in a block (e.g., due to rotations, several moving objects in a block, etc.). S_i would be zero if all the pixels in the block moved with the same velocity, but in practice S_i is approximately zero only for small block size B and increases with larger B. In fact, in Section 6.1 of the Appendix we show that assuming a separable, first-order autoregressive model for the ideal motion vector field of F, S_i can be approximated by

$$S_i \approx 6\,\sigma_V^2 \ln(1/a) A_i B, \tag{6}$$

where σ_V^2 and a are the variance and correlation coefficient of components of the ideal motion vectors (same for the X and Y components), respectively. A fairly accurate estimate of σ_V^2 can be computed from the empirically measured motion vectors at each block, but an accurate estimate of a requires finding the ideal motion vector at every pixel, also known as optical flow, which is very complex (c.f., Barron *et al.*[18]). In practice, a can be fixed for a large variety of scenes, and based on the experiments in Guillotel and Chevance's work,[14] we fix $a = 0.998$[l]. Combining (1), (2), (3), (4), (5), and (6) yields the following approximation to the total difference frame rate R_D for the low distortion case:

$$R_D \approx \frac{e}{Q^2 \ln 2}\left(\Delta^2 T_A + 6\,\sigma_V^2 \ln(1/a) T_A B + \frac{Q^2}{12} + \mu\right), \tag{7}$$

where

$$T_A = \frac{1}{P}\sum_{i=1}^{P/B^2} A_i = \frac{1}{12P}\sum_{(x,y)\epsilon F} \left((F(x,y) - F(x+1,y))^2 + (F(x,y) - F(x,y+1))^2\right) \tag{8}$$

is a measure of the texture (per pixel) in the current frame F (independent of the block size value B), and

$$\mu = \frac{1}{P}\sum_{i=1}^{P/B^2} C_{0,i} \tag{9}$$

is the average interframe noise (per pixel). Observe that μ is not a function B, because recall that the $C_{0,i}$'s are noise energies caused by light changes, occlusions, etc., which will add to the same total μP regardless of the value of block size. (The value of μ will not be needed.)

2.2 Motion vector rate R_M

The motion vectors are losslessly encoded by DPCM with entropy coding. Let $n_i = (n_{x,i}, n_{y,i})$ be the DPCM prediction error for the motion vector at the ith block. Note that the $n_{x,i}$'s and $n_{y,i}$'s are integer multiples of Δ, the pixel accuracy with which the motion vectors are found. We model the $\{n_{x,i}, n_{y,i}\}_{i=1}^{P/B^2}$ values as Δ-quantized outcomes of a continuous Laplacian random variable $\mathcal{N}$ with variance $\beta_{\mathcal{N}}^2$. Then, the total expected rate invested in motion is:

$$R_M \approx \frac{1}{P}\sum_{i=1}^{P/B^2} \left(2h(\mathcal{N}) - \log_2 \Delta^2\right), \tag{10}$$

[l] Note that the correlation coefficient of the ideal motion vectors is significantly higher than that of the pixel values - typically 0.9

with

$$h(\mathcal{N}) = \frac{1}{2}\log_2 2e^2\beta_{\mathcal{N}}^2, \qquad (11)$$

the differential entropy of $\mathcal{N}$. The approximation in (10) is obtained, like the low distortion case in (1), by relating the entropy of the $n_{x,i}$'s and $n_{y,i}$'s to the differential entropy of a Laplacian.[15]

Clearly, the prediction error variance $\beta_{\mathcal{N}}^2$ will increase with B because the correlation of adjacent motion vectors decreases when the vectors are further apart. In fact, in Section 6.2 of the Appendix we show that assuming a separable, first-order autoregressive model for the computed motion vectors and neglecting small terms:

$$\beta_{\mathcal{N}}^2 \approx 2\,\tilde{\sigma}_V^2 \ln(1/\tilde{a})B, \qquad (12)$$

where $\tilde{\sigma}_V^2$ and $\tilde{a}$ are the average variance and correlation coefficient, respectively, measured for the X and Y components of the computed motion vectors. It is important to observe that the ideal motion field (which we previously modeled also with an AR process) and the field of computed motion vectors have different statistics. In practice, our final results are not sensitive to the differences between the variance of the ideal motion field σ_V^2 and that of the computed field $\tilde{\sigma}_V^2$, and we set $\sigma_V^2 = \tilde{\sigma}_V^2$. But the correlation factor a of the ideal motion field (which we fixed to 0.998 based on accurate motion estimates[14]) may be significantly larger than that of the computed field (which we find with classical block matching[1]). Because of this, we will estimate $\tilde{a}$ from the computed motion vectors, and since we will not compute the motion vectors at every pixel but at every block, we will use equation (23) with the appropriate value of B.

Now, combining (10), (11), and (12), we obtain an expression for the motion vector rate:

$$R_M \approx \frac{1}{P}\sum_{i=1}^{P/B^2}\left(\log_2(4e^2\sigma_V^2\ln(1/\tilde{a})B) - \log_2\Delta^2\right) = \frac{1}{B^2}\log_2\frac{4e^2\sigma_V^2\ln(1/\tilde{a})B}{\Delta^2}. \qquad (13)$$

2.3 Total rate R

Finally, we sum the motion rate R_M (10) and the difference frame rate R_D (7), and obtain the following approximation to the total rate for encoding the current frame:

$$\begin{aligned} R &= R_D + R_M \\ &\approx \frac{e}{Q^2\ln 2}\left(\Delta^2 T_A + 6\,\sigma_V^2\ln(1/a)T_A B + \frac{Q^2}{12} + \mu\right) + \frac{1}{B^2}\log_2\frac{4e^2\sigma_V^2\ln(1/\tilde{a})B}{\Delta^2}. \end{aligned} \qquad (14)$$

Observe that (14) indicates the effect of block size B and other interesting parameters on the encoding rate R. As expected, the difference frame rate R_D increases with larger B, because the difference frame energy due to non-uniform motion, "$6\,\sigma_V^2\ln(1/a)T_A B$", increases with B. Also, the motion rate R_M typically increases with smaller B, since reducing the block size increases the number of motion vectors, as indicated by "$1/B^2$". However, reducing B increases the correlation between the motion vectors and fewer bits per vector are needed, as shown by "$\log_2(4e^2\sigma_V^2\ln(1/\tilde{a})B)/\Delta^2$". In fact, note that if "$\sigma_V^2\ln(1/\tilde{a})$" was very small (i.e., very large motion coherence), the correlation effect would dominate and reducing block size would also reduce R_M (e.g., $\tilde{a} = 0.99$, $\Delta = 1$, and $\sigma_V^2 < 1$).

Not surprisingly, decreasing Δ (higher motion vector accuracy) increases the motion rate R_M, but decreases the difference frame rate R_D since the prediction for the current frame is improved with more accurate motion vectors. Finally, R_D increases with higher frame texture T_A, coding distortion Q (or $Q^2/12$ in MSE), and interframe noise μ.

3 OPTIMIZATION

3.1 The optimal motion vector accuracy Δ^*

For a fixed block size B and distortion Q, we find that the motion vector accuracy Δ^* that minimizes the total frame rate R (14) is

$$\Delta^* = L\left(\frac{D}{T_A B^2}\right)^{\frac{1}{2}}, \qquad (15)$$

where $L = \sqrt{12/e}$ is a constant and $D = Q^2/12$ is the approximate MSE distortion of the encoded frame. Equation (15) indicates that the motion vectors must be encoded more accurately (i.e., Δ smaller) for larger block size. Also, higher motion accuracy is necessary in frames with larger texture T_A, and lower accuracy is needed at higher levels of compression (higher D).

3.2 The optimal block size B^*

For a fixed motion accuracy Δ and distortion Q, we find that the block size B^* that minimizes R (14) satisfies

$$K_1 B^{*3} = 2\log_2 B^* + K_2,$$

where

$$K_1 = \frac{T_A \sigma_V^2 \ln(1/a)}{D}\frac{e}{2\ln 2}, \qquad K_2 = 2\log_2 \frac{\sigma_V^2 \ln(1/\tilde{a})}{\Delta^2} + 4 + 3\log_2 e. \qquad (16)$$

The dependence of B^* on Δ, T_A, D, σ_V^2, a and $\tilde{a}$ is explained in detail in the next section.

4 EXPERIMENTAL RESULTS

We present results of the performance of our video coder and of our formulas on the well-known video sequence "caltrain" (a scene with a moving train and a calendar) with quantizer step size Q=20 (the PSNR is approximately 33 dB). The frame resolution was 512 x 400 pixels and only the gray-level components were encoded. We used the full-search block-matching method with the minimum absolute difference criteria[1] to compute the motion vectors with pixel and subpixel accuracy for Δ's in the set $S_\Delta = \{2, 1, 1/2, 1/4, ..., 1/32\}$ (subpixel accuracy was computed using bilinear interpolation on the pixel position with the best match), and the maximum interframe displacement was 8 pixels. The first frame was intracoded by a simple uniform scalar quantizer with quantization step Q and each of the following frames were predicted by their respective (encoded) previous frames.

In Figure 1, the continuous line shows the total empirical rate R for encoding the second frame of "caltrain" using block size $B = 16$ (16×16 pixels/block) for different values of Δ. The two dashed lines show $R(\Delta)$ but for $B = 4$ and $B = 8$. The "o" signs on the Δ axis show our prediction Δ^* (15) for where the minimum of R occurs. From left to right, the "o"'s correspond to the $B = 4$, $B = 8$, and $B = 16$ rate curves, respectively. The "*" signs plot the rate obtained when the Δ^*'s are rounded to the nearest Δ in S_Δ.

In Figure 2, the continuous line corresponds to "$K_1 B^3$" (see (16)) for the second frame of "caltrain". We measured $T_A = 30.732$, and estimated $\sigma_V^2 = 4.095$ and $\tilde{a} = 0.837$ from the computed motion vectors with $B = 4$ and $\Delta = 1/4$ ($a = 0.998$). The dashed lines correspond to "$2\log_2 B + K_2$" for different values of Δ. The points where the dashed lines cross the continuous line are the values B^* for the respective Δ's. Observe from this plot and from (16) that the best block sizes B^* decrease with larger frame texture T_A and smaller distortion D (larger

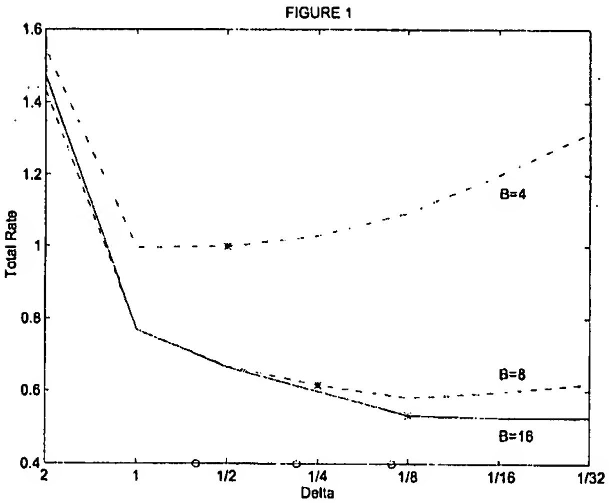

Figure 1: Total empirical rate R vs. motion vector accuracy Δ, for fixed block size $B = 4$ (4×4 pixels per block), $B = 8$ and $B = 16$. The "o" signs on the Δ axis show our prediction Δ^* (15) for where the minimum of R occurs. From left to right, the "o"'s correspond to the $B = 4$, $B = 8$, and $B = 16$ rate curves, respectively. The "$*$" signs plot the rate obtained with our optimal motion accuracies for each B case (when the Δ^*'s are rounded to the nearest Δ in S_Δ).

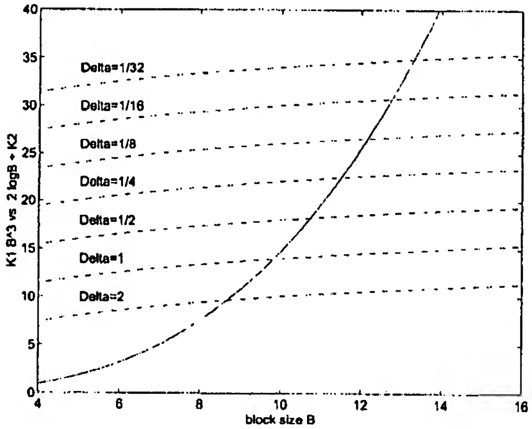

Figure 2: $K_1 B^3$ (solid line) vs. $2 \log_2 B + K_2$ (dashed lines), for different motion vector accuracies Δ (see equation (16)). The points where the dashed lines cross the continuous line are the optimal block size values B^* for the respective Δ's.

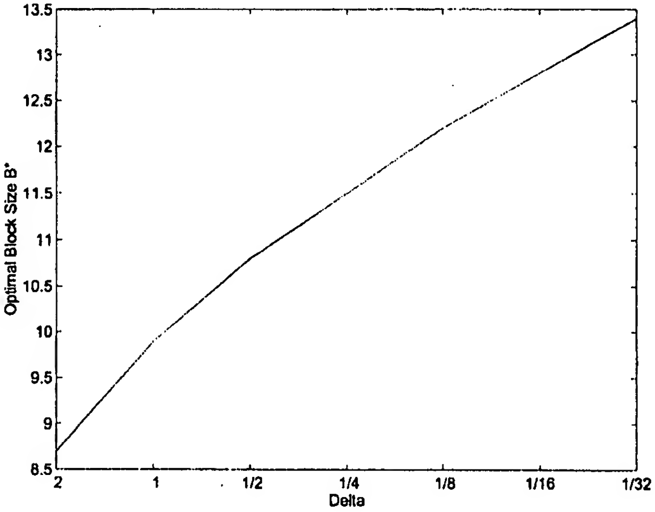

Figure 3: Optimal block size B^* vs. motion accuracy Δ for "caltrain" video sequence.

K_1), and that the B^*'s increase with higher motion vector accuracy (larger K_2). Also, note from (16) that large motion coherence (small σ_V^2 and large a, $\tilde{a}$) decreases both K_1 and K_2. Typically, the effect of K_1 will dominate and the crossings will occur at larger values of B. In other words, the optimal B^*'s usually increase for more coherent motion fields. But in the rare case that "$\sigma_V^2 \ln(1/\tilde{a})/\Delta^2$" is extremely small (e.g., $\tilde{a} = 0.99$, $\Delta = 1$, and $\sigma_V^2 < 1$), then the effect of K_2 would dominate and decrease the B^*'s.

In Figure 3, the continuous line shows the optimal block size B^* plotted versus motion accuracy Δ for the second frame of "caltrain". The B^*'s (the line crossings in Figure 2) are found using a simple search procedure in (16).

In Figure 4 the continuous line shows the total empirical rate R for encoding the second frame of "caltrain" using pixel accurate motion vectors (Δ=1) for different values of block size B. The dashed, dash-dotted, and dotted lines show $R(B)$ but when the motion vectors are found with $\Delta = 1/2$, $\Delta = 1/8$, and $\Delta = 1/32$, respectively. The "o" signs on the B axis show our prediction B^* for where the minimum of R occurs. From left to right, the "o"'s correspond to the $\Delta = 1$, $\Delta = 1/2$, $\Delta = 1/8$ and $\Delta = 1/32$ rate curves, respectively. The "$*$" signs plot the rate obtained with our optimal block size when the B^*'s are rounded to the nearest B in the set of interest $S_B = \{4, 8, 16, 32\}$.

Figure 5 is like Figure 1 but for the third frame of "caltrain". Results for this and other frames of "caltrain" are very similar to those of Figures 1, 2, 3, and 4 because the texture and the motion vector statistics remain approximately constant in the entire sequence.

The results suggest that our equations work well. Our optimal B^*'s appear to be close to the minimum of the rate curves in Figure 4. The exact value of the B^*'s that minimize the underlying continuous rate curves is not known, but when we round our B^* to the closest B in the set of interest S_B, the rate that we obtain is always the minimum. Even though block sizes $B = 8$ and $B = 16$ perform similarly for "caltrain" with pixel ($\Delta = 1$) and half-pixel ($\Delta = 1/2$) accurate motion vectors at the considered level of distortion (Figure 1), our method indicates that $B = 8$ should be chosen for those lower motion accuracies (recall Figure 4). This information is important because smaller B's are preferred to reduce blocking artifacts from the block-based prediction. Finally, our optimal Δ^*'s also appear to be close to the minimum of the rate curves in Figures 1 and 5. Again, the exact

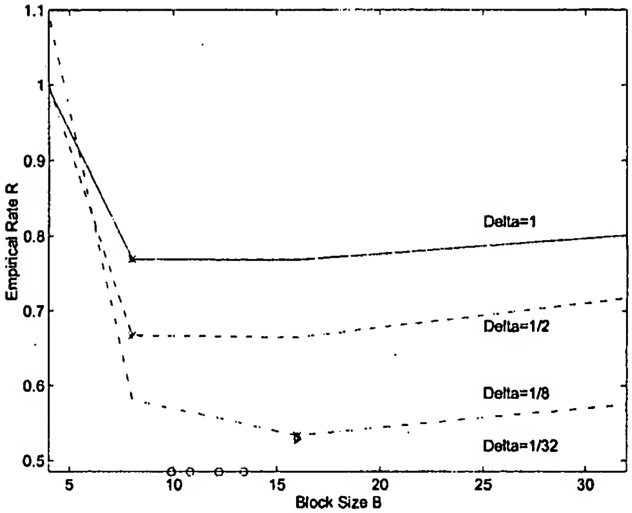

Figure 4: Total empirical rate R vs. block size B, for fixed motion accuracy Δ. The "o" signs on the B axis show our prediction B^* for where the minimum of R occurs. From left to right, the "o"'s correspond to the $\Delta = 1$ (pixel accurate vectors), $\Delta = 1/2$, $\Delta = 1/8$ and $\Delta = 1/32$ rate curves, respectively. The "$*$" signs plot the rate obtained with our optimal block size (when the B^*'s are rounded to the nearest B in S_B) .

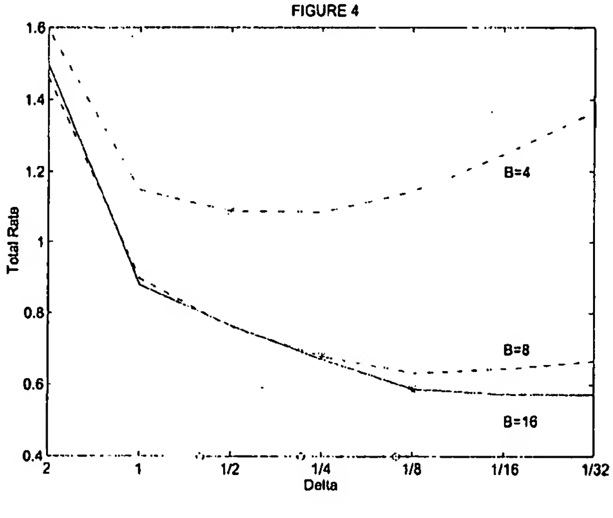

Figure 5: Total empirical rate R vs. motion vector accuracy Δ. Like Figure 1 but for the third frame of "caltrain"

value of the Δ's that minimize the underlying continuous rate curves is not known, but when we round our Δ^*'s to the closest Δ in the set of interest S_Δ, the rate that we obtain is either the minimum or very close to the minimum.

5 CONCLUSIONS

In a classical block-based video coder, our simple techniques find at every frame the optimal block size B^* for a given motion vector accuracy Δ, and the optimal Δ^* for a given B. Our formulas and results show the effects of the key characteristics of the video scene on B^* and Δ^*. Specifically, they indicate that the block size should be smaller for higher textured frames and when the motion vectors are encoded with low accuracy. Also, the block size should be larger for larger coding distortions, and when there is motion coherence in the frame (i.e., small motion vector variance and large motion vector correlation). Consequently, it is not surprising that Docef and Smith[13] found $B = 2$ to be the best block size for ultrasound video, since the structure and texture components of those sequences have high textured frames and low motion coherence. Conversely, our equations indicate that the motion vectors should be encoded more accurately for larger blocks and when there is higher texture in the frame, and less accurately when the level of compression is high.

Some of the above relationships have been noted by other researchers and have been used heuristically in practice. Other relationships are more confusing and not so well understood. For example, it is not obvious whether larger motion coherence should lead to decreasing the block size in order that the motion vector coder can take advantage of the motion vector correlation, or to larger blocks so there will be fewer motion vectors to encode without substantially increasing the difference frame rate. In fact, our equations model both effects (c.f. see (14)) and show that the second interpretation is typically the correct one, except in the rare case when "$\sigma_V^2 \ln(1/\tilde{a})$" is very small (i.e., very large motion coherence). Not surprisingly, Martin *et al.*[10] designed fairly good variable block-size motion fields for video coders by heuristically joining blocks with coherent motion (similar motion vectors) into larger blocks.

We believe that our analysis is also a good first step toward understanding and deriving simple procedures to find the optimal motion resolution for object-based and variable block-size video coders, which is our ongoing work. Finally, note that many other block-based video coders use difference frame encoders that are more sophisticated than our uniform scalar quantization and entropy coding. For example, MPEG[2] uses a block-by-block DCT, which gives better rate-distortion performance and also exploits the response of the human visual system. In this work, our coder does not intend to compete with these more advanced video coders but to explore the block size problem in a basic case. The rates of those more advanced coders are expected to be lower than ours by a constant, but the relative differences between the rate curves modes are expected to remain roughly the same, as was found in previous work.[16,17]

6 APPENDIX

6.1 Prediction error caused by nonuniform motion in a block

In this section, we model the effect of having different pixel motions in a block on the energy of the block's prediction error. To simplify this discussion, we consider one-dimensional, continuous frames. Let $P(x)$ be a block of length B (in pixel units) in the current frame and $\tilde{P}_V(x)$ the block in the previous frame indicated by P's motion vector V. For convenience, assume that $P(x)$ and $\tilde{P}_V(x)$ take (intensity) values for $x \in [0, B]$ and are zero outside that interval. Now, let $v(x)$ be the ideal motion vector field for the pixels in P, so that $v(x)$ is constant at each pixel, i.e., $v(x) = v_i$ for $x \in [i, i+1)$. Here, we assume there is no prediction noise other than that caused by nonuniform motion, and therefore if all the pixels in P moved with the same velocity, $v(x) = V$, $P(x)$ and $\tilde{P}_V(x)$ would be identical, $P(x) = \tilde{P}_V(x)$. But since typically different pixels have different velocities within a block, we model $\tilde{P}_V$ as follows:

$$\tilde{P}_V(x) = P(x + (V - v(x))). \tag{17}$$

Next, we model the motion field $v(x)$ as a realization of a random process $V(x)$ such that $V(x) = V_i$ for $x \in [i, i+1)$, where V_i is the ith variable of a random sequence, and assume that $v(x)$ takes the value of the block's motion vector in the middle of the block. Our goal is to find a simple expression for:

$$S = E[\int_0^B (P(x) - P(x + (V(B/2) - V(x))))^2 dx]. \tag{18}$$

Since $v(B/2) - v(x)$ is small, we expand $P(x + (v(B/2) - v(x)))$ in a Taylor series and obtain:

$$\begin{aligned} S &\approx E[\int_0^B (P'(x)(V(B/2) - V(x)))^2 dx] = \int_0^B P'^2(x) E[(V(B/2) - V(x))^2]dx \\ &= \sum_{i=0}^{B} E[(V_{B/2} - V_i)^2] \int_{x=i}^{i+1} P'^2(x)dx, \end{aligned} \tag{19}$$

with $P'(x)$ the derivative of $P(x)$. In Section 6.2 we show that assuming a highly-correlated, autoregressive (AR) model for the random sequence V_i,

$$E[(V_{B/2} - V_i)^2] \approx 2\sigma_V^2 \ln(1/a)|B/2 - i|, \tag{20}$$

where σ_V^2 is the variance of V_i and a is the AR coefficient. Combining (19) and (20), and after some manipulations, we find the following approximation for S:

$$\begin{aligned} S &\approx \frac{1}{2}\sigma_V^2 \ln(1/a) B \int_0^B P'^2(x)dx \approx \frac{1}{2}\sigma_V^2 \ln(1/a) B \sum_{i=0}^{B} (P(i) - P(i+1))^2 \\ &= 6\sigma_V^2 \ln(1/a) B\, A, \end{aligned} \tag{21}$$

where $A = (1/12)\sum_i (P(i) - P(i+1))^2$. In the two-dimensional case, we obtain the same result for S but A is like (4).

6.2 The variance of the difference between AR random variables

Here, we find an approximate expression for the variance of the difference between random variables of a zero-mean, first-order autoregressive (AR) process. Specifically, consider the following AR random sequence:

$$V_j = aV_{j-1} + n_j, \tag{22}$$

where V_j is the jth random variable of the sequence, a is the correlation coefficient ($|a| < 1$), and $\{n_j\}$ is a sequence of iid random variables with zero mean. Our goal is to obtain a simple expression for $E[(V_j - V_{j-B})^2]$, where B is the distance between random variables V_j and V_{j-B} in the sequence. It is easy to show that:

$$E[(V_j - V_{j-B})^2] = E[V_j^2] + E[V_{j-B}^2] - 2E[V_j V_{j-B}] = 2\sigma_V^2 - 2\sigma_V^2 a^{|B|}, \tag{23}$$

with σ_V^2 the variance of V_j (same for all j). To simplify (23), we expand $a^{|B|}$ in a Taylor series:

$$a^{|B|} = 1 + B\ln a + \frac{(B\ln a)^2}{2} + \frac{(B\ln a)^3}{3} + \ldots \tag{24}$$

If a is close to 1, observe that $a \approx 1 + B \ln a$, and then

$$E[(V_j - V_{j-B})^2] \approx 2\sigma_V^2 B \ln(1/a). \tag{25}$$

Hence, the variance of the difference between random variables V_j and V_{j-B} of a highly correlated AR random sequence (a close to 1) is approximately linear with B, as expressed in (25).

7 REFERENCES

[1] A. Netravali and B. Haskell, *Digital pictures. Representation and compression*, Plenum Press, 1988.

[2] D. Le Gall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, Vol. 34, pp. 47-58, Apr. 91.

[3] ITU-T Recommendations H.261, *Video codec for audiovisual services at px64 kbits.*

[4] ITU-T/SG15, *Video codec test model, TMN5*, Telenor Research, Jan. 95.

[5] P. Strobach, "Tree-structured scene adaptive coder," IEEE Trans. Communications, Vol. 38, Apr. 90.

[6] G. J. Sullivan and R.L. Baker, "Rate-distortion optimized motion compensation for video compression using fixed or variable size blocks," Proc. GLOBECOM, pp.85-90, Nov. 91.

[7] F. Moscheni, F. Dufaux and H. Nicolas, "Entropy criterion for optimal bit allocation between motion and prediction error information," *Proc. VCIP*, pp. 235-242, Nov. 93.

[8] B. Girod, "Motion-compensating prediction with fractional-pel accuracy," *IEEE Trans. Commun.*, Vol. 41, pp. 604-612, Apr. 93.

[9] B. Girod, "Rate-constrained motion compensation," *Proc. VCIP*, pp. 1026-1034, Chicago, Sept. 94.

[10] G.R. Martin, R.A. Packwood, I. Rhee, "Variable size block matching motion estimation with minimal error," *Proc. SPIE Dig. Video Compr.*, pp. 324-333, San Jose, Jan-Feb 96.

[11] G.M. Schuster and A.K. Katsaggelos, "A video compression scheme with optimal bit allocation between displacement vector field and displaced frame difference," *Proc. ICASSP*, Atlanta, May 96.

[12] M. Sipitca and V. Madisetti, "Lossy techniques for motion vector encoding," *Proc. VCIP*, pp. 387-393, Orlando, Mar. 96.

[13] A. Docef and M. Smith, "Application of motion compensated prediction to coding of ultrasound video," *Proc. VCIP*, pp. 811-817, Orlando, Mar. 96.

[14] P. Guillotel and C. Chevance, "Comparison of motion vector coding techniques," *SPIE Image and Video Compr.*, pp. 1594-1604, Feb. 94.

[15] T.M. Cover and J.M Thomas, *Elements of Information Theory* John Wiley & Sons, Inc., 91.

[16] J. Ribas-Corbera and D.L. Neuhoff, "On the optimal motion vector accuracy for block-based, motion-compensated video coders," *Proc. SPIE Dig. Video Compr.*, pp. 302-314, San Jose, Jan-Feb 96.

[17] J. Ribas-Corbera and D.L. Neuhoff, "Reducing rate/complexity in video coding by motion estimation with block adaptive accuracy," *Proc. VCIP*, pp. 615-624, Orlando, Mar. 96.

[18] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt, "Performance of optical flow techniques," *Proc. IEEE Comp. Vision and Pattern Recog.*, pp. 236-242, 1992.

On the Optimal Motion Vector Accuracy for Block-Based Motion-Compensated Video Coders

Jordi Ribas-Corbera and David L. Neuhoff

Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109, USA
jordi@eecs.umich.edu, neuhoff@eecs.umich.edu

ABSTRACT

In block-based motion-compensated video coding, a fixed-resolution motion field with one motion vector per image block is used to improve the prediction of the frame to be coded. All motion vectors are encoded with the same fixed accuracy, typically 1 or 1/2 pixel accuracy. In this work, we explore the benefits of encoding the motion vectors with other accuracies, and of encoding different motion vectors with different accuracies within the same frame. To do this, we analytically model the effect of motion vector accuracy and derive expressions for the encoding rates for both motion vectors and difference frames, in terms of the accuracies. Minimizing these expressions leads to simple formulas that indicate how accurately to encode the motion vectors in a classical block-based motion-compensated video coder. These formulas also show that the motion vectors must be encoded more accurately where more texture is present, and less accurately when there is much interframe noise. We implement video coders based on our analysis and present experimental results on real video frames. These results suggest that our equations are accurate, and that significant bit rate savings can be achieved when our optimal motion vector accuracies are used.

Keywords: video coding, motion estimation, motion vector accuracy, adaptive motion vector accuracy.

1 INTRODUCTION

In motion-compensated video coding, motion vectors are used to improve the prediction of the frame to be encoded.[1–4] These motion vectors must themselves be encoded with bits. The benefit of such an investment is that the sum of the motion rate $\bar{R}_M$ (the number of bits per pixel to encode the motion vectors) plus the difference frame rate $\bar{R}_D$ (the number of bits per pixel to encode the motion-compensated difference frame) can be significantly less than the rate (in bits per pixel) required to encode the difference frame obtained without motion compensation. The widespread success of motion compensation is testimony to the value of this strategy.

The motion rate $\bar{R}_M$ depends on the number of motion vectors (equivalently, the spatial resolution of the motion vectors) and the accuracies with which motion vectors are encoded (e.g., to one, one-half, one-quarter or one-eighth pixel accuracy). Increasing either will decrease the difference frame energy and, if the coder keeps distortion constant, will also decrease the difference frame rate $\bar{R}_D$ and, up to a point, the total encoding rate $\bar{R} = \bar{R}_M + \bar{R}_D$.

In this paper we focus on quantifying the dependence of the total encoding rate on the accuracies with which motion vectors are encoded, assuming distortion is held constant. Specifically, for a given set of motion vectors, we find simple approximate analytical expressions for how the motion rate $\bar{R}_M$ and the difference frame rate $\bar{R}_D$ depend on the accuracies. And we use them to optimize the performance of video coders.

We focus on classical block-based video coders with one motion vector per image block. (The method could be applied equally well to variable-block sized video coders.) Each motion vector is encoded by uniform scalar quantization of the differences between its x and y components and those of the motion vector of the previous block, followed by entropy coding. The quantizer step size determines the accuracy of the motion vector — smaller step sizes result in larger accuracy and larger motion rate. We find simple approximate expressions for the resulting motion rate $\bar{R}_M$ and the difference frame rate $\bar{R}_D$. The key step in developing the latter is the derivation of a simple quadratic expression for the energy of the resulting difference frame, in terms of the quantizer step sizes. Then, assuming that difference frame pixels have a Laplacian distribution, an expression for the difference frame rate follows easily for a coder that uses uniform scalar quantization of the difference frame pixels followed by entropy coding. (The method is also tested with DCT coding of the difference frames.)

In most video coders, all motion vectors are encoded with the same accuracy,[1-11] and the choice of accuracy — typically 1 or 1/2 pixel accuracy[1-5] — and resulting motion rate are based on heuristics and empirical experiments. In contrast, the expressions we derive for $\bar{R}_M$ and $\bar{R}_D$ are simple enough that for each video frame an encoder can dynamically optimize the choice of the motion vector accuracy (quantizer step size) for each block, so as to minimize the total rate for a given level of distortion in the encoded frames. The resulting expressions indicate that motion vectors must be encoded more accurately where there is more texture and less accurately where the prediction is corrupted by phenomena such as camera noise, occlusions or distortion from the encoding of previous frames (when compression is high). Some of these relationships have been previously observed and heuristically used in the design of video coders.[5,8,9,11]

Based on our techniques, we implement video coders with two types of difference frame coding: uniform scalar quantization with entropy coding (for which our analysis is based) and DCT coding (for which the motion accuracies determined for the previous case can also be used).

Our results include rate-distortion curves for the typical motion vector accuracies (1 and 1/2 pixel) and for our optimal accuracies, and rate-accuracy curves for several levels of distortion. These results suggest that our equations are accurate, and that significant bit rate savings can be achieved if our optimal motion vector accuracies are considered. For example, adapting the motion vector accuracy to individual blocks with our procedure results in up to 17 per cent bit rate savings with respect to encoding all motion vectors with the same 1/2 pixel accuracy (at a peak SNR of 31 dB). Finally, we discuss potential applications of this work to other video coders and to block adaptive motion estimation.

This work is an extension of previous work of the authors[13] that considered the optimization of motion vector accuracies for lossless video coders. There are several key differences. An obvious one is the inclusion of the lossy quantizer in the difference frame encoder, which necessitated a change in the expressions for $\bar{R}_D$ in terms of the difference frame energy. Another is the DPCM encoding of the motion vectors, in contrast to the simpler scalar quantization in the lossless case, which is not adequate for lossy coding. Due to this, a substantially different technique is needed to estimate the difference frame energy. Finally, in the lossy case, the motion rate is a larger fraction of the total and more benefit results from the optimization of the motion accuracies.

Other researchers have also addressed the effect of motion vector accuracy on difference frame energy and total rate. Girod[7,8] found expressions for the difference frame energy as a function of motion vector error distribution, and the spectrum of the current frame and that of the interframe noise. He reached some interesting conclusions from this analysis, but unfortunately it cannot be used to adapt the accuracies on a block-by-block basis, and even for entire an frame the resulting expressions are too complex to be used to analytically derive the optimal motion vector accuracy. Vandendorpe *et al.*[10] presented work similar to that of Girod,[8] but for interlaced frames. Ito and Farvardin[11] performed a simple analysis of difference frame energy in the context of subband coding, which

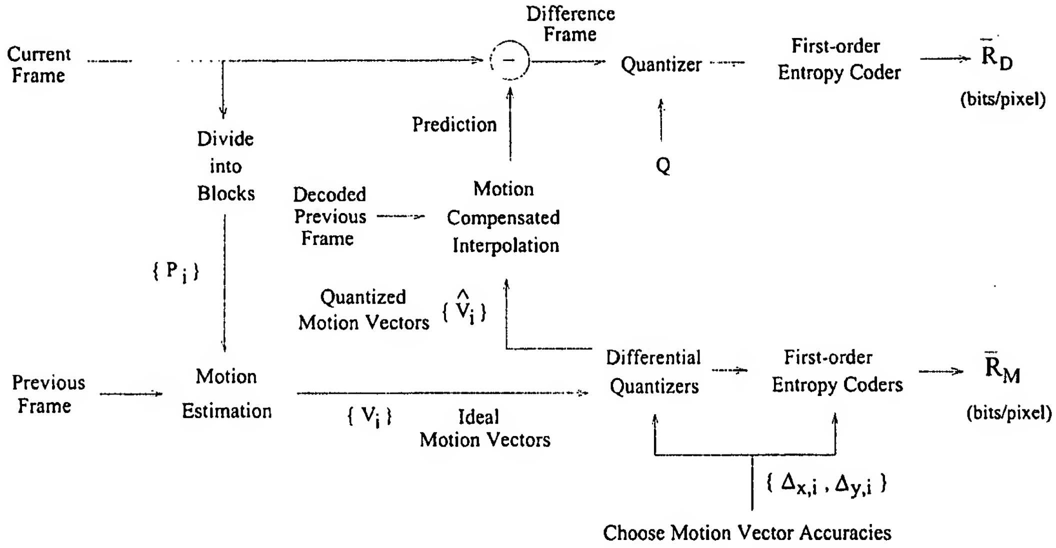

Figure 1: "Coder 1"

was sufficient to conclude that the motion vectors of higher frequency subbands require more accurate encoding.

In another interesting work, Moscheni *et al.*[12] use an entropy formula, similar to ours, to predict the rate of the difference frame encoder based on measurements of the difference frame energy, and then decide how to allocate motion bits in a hierarchical motion compensation scheme. Aside from the fact that we do not consider hierarchical motion compensation, our method differs in that we model the difference frame energy and analytically derive motion allocations, whereas they had to find them by trial and error, and we jointly optimize the motion rates, whereas they perform a suboptimal one-rate-at-a-time optimization.

The effect of the spatial resolution of the motion vectors, which is not considered in the present paper, has been addressed in some of the previously cited work[5,9,12] and other papers.[15]

2 A BLOCK-BASED VIDEO CODER

In a typical block-based video coder a prediction for the frame to be encoded is formed by subdividing the frame into blocks $\{P_i\}$ (e.g., 8×8 pixels per block), finding the *motion vector* V_i for each P_i that displaces it to match a block in the previous frame[†], and then displacing the blocks in the decoded previous frame according to the quantized versions of the motion vectors, $\{\hat{V}_i\}$. (If the components of the motion vectors are not integer multiples of $\Delta_T \triangleq 1$, the pixel spacing, the intensity values for the blocks in the previous frame are obtained by interpolation.) Subtracting the predicted frame from the actual frame yields a *difference frame*. This process is illustrated in Figure 1.

Ordinarily, the motion vectors are found by a simple motion estimation procedure such as block-matching,[1] which computes the motion vectors $\{V_i\}$ with a chosen accuracy, and hence a separate quantization step is not actually needed (i.e., $V_i = \hat{V}_i$ for all i). In this study, however, it is assumed that the motion vectors are computed

[†] Our analysis can also apply to backward prediction or interpolative prediction.

to some very high accuracy Δ_0 (i.e., the components of V_i are integer multiples of Δ_0, where $\Delta_0 \ll \Delta_T$), and then quantized to some coarser accuracy $\Delta > \Delta_0$ to obtain the $\hat{V}_i$'s.

Figure 1 shows the block diagram for "Coder 1", a simple block-based video coder. In "Coder 1", the difference frame pixels are encoded by a uniform scalar quantizer with quantization step Q followed by a first-order entropy coder where the code used is adapted to that particular difference frame. (Note that the image pixels take values in $\{0, ..., 255\}$. When $Q = 1$, the pixels are losslessly encoded, and when $Q > 1$, they are lossy encoded.) Such a coder is an adequate choice for this study because of the low correlation between the pixel values in the difference frame.[7,8,12] Letting $\hat{p}_Q(d)$ denote the frequency of the value d in the Q-quantized difference frame, the expected rate (number of bits/pixel) produced when encoding a particular difference frame is

$$\bar{R}_D \approx -\sum_d \hat{p}_Q(d) \log_2 \hat{p}_Q(d) \triangleq H(\hat{p}_Q). \tag{1}$$

The motion vectors are lossy encoded by DPCM with variable-length coding. This technique is popular in coding with a fixed-resolution motion field, since it has been shown to perform well in practice.[2,4,6] In our scheme, each motion vector is predicted by the quantized version of the previous vector, in scan order. Let $n_i = (n_{x,i}, n_{y,i})$ be the prediction error for the motion vector at the ith block, with $n_{x,i}$ the horizontal or x component of the prediction error and $n_{y,i}$ the vertical or y component. We quantize $n_{x,i}$ and $n_{y,i}$ using uniform scalar quantizers with quantization steps $\Delta_{x,i}$ and $\Delta_{y,i}$, respectively, and encode the results using a different entropy coder for each value of Δ. In the classical video coding approach all motion vectors are encoded with the same accuracy $\Delta = \Delta_{x,i} = \Delta_{y,i}$. In our new approach, the Δ's are permitted to differ from block to block, and $\Delta_{x,i}$ can be different from $\Delta_{y,i}$. Specifically, there is a collection of such codes $\{C_{\delta_1}, C_{\delta_2}, \ldots\}$, and code C_{δ_j} is used to encode the quantization of $n_{x,i}$ whenever $\Delta_{x,i} = \delta_j$, and similarly for $n_{y,i}$. Let us assume that the $\{n_{x,i}, n_{y,i}\}$ values are outcomes of a random variable $\mathcal{N}$ with variance $\sigma^2_{\mathcal{N}}$. Then, the total expected rate invested in motion is

$$\bar{R}_M \approx \frac{1}{P} \sum_{i=1}^{N} \left(H(\mathcal{C}_{\Delta_{x,i}}) + H(\mathcal{C}_{\Delta_{y,i}}) \right) \tag{2}$$

where P is the number of pixels per frame, N is the number of blocks per frame, and $H(\mathcal{C}_\Delta)$ is the entropy of the random variable $\mathcal{N}$ quantized with Δ accuracy. For $\Delta \ll \sigma_{\mathcal{N}}$ it can be shown[14] that

$$H(\mathcal{C}_\Delta) \approx h(\mathcal{N}) - \log_2 \Delta, \tag{3}$$

where $h(\mathcal{N})$ is the differential entropy of $\mathcal{N}$. Then, the total expected rate invested in motion is

$$\bar{R}_M \approx \frac{1}{P} \sum_{i=1}^{N} (2h(\mathcal{N}) - \log_2 \Delta_{x,i}\Delta_{y,i}). \tag{4}$$

Of course, the Δ's must also be encoded. However, we will not count the rate for this, because it is usually negligible compared to $\bar{R}_M$ and $\bar{R}_D$. Finally, the expected total rate to encode the current frame is

$$\bar{R} = \bar{R}_D + \bar{R}_M \approx H(\hat{p}_Q) + \frac{1}{P} \sum_{i=1}^{N} (2h(\mathcal{N}) - \log_2 \Delta_{x,i}\Delta_{y,i}). \tag{5}$$

Observe that making the Δ's smaller increases the motion rate $\bar{R}_M$ but improves the prediction for the current frame, which decreases the energy in the difference frame and, if the distortion of the coder is fixed, the difference frame rate $\bar{R}_D$. Hence, when the Δ's are small, $\bar{R}_M$ is large and $\bar{R}_D$ is small, and when they are large, the opposite occurs. On the other hand, the Δ's have little effect on the overall mean squared distortion of "Coder 1", because this is well approximated by $Q^2/12$. This suggests that Q should be fixed to achieve a desired level of distortion and that for each frame, the encoder should dynamically choose the Δ's to minimize the total rate $\bar{R}$ for that frame, as expressed in (5).

3 THE OPTIMAL MOTION VECTOR ACCURACIES

To minimize the expression for $\bar{R}$ in (5) for a given Q, the encoder must be able to predict $H(\hat{p}_Q)$ in terms of Q and $\{\Delta_{x,1}, \Delta_{y,1}, ..., \Delta_{x,N}, \Delta_{y,N}\}$. (Note that the constant value of $h(\mathcal{N})$ does not affect the minimization.)

To predict $H(\hat{p}_Q)$, we assume that $\hat{p}_Q(d)$ has a discrete Laplacian form with variance σ^2 and use the approximate formula

$$H(\hat{p}_Q) \quad \approx \quad \begin{cases} \log_2 \sqrt{2}e\sigma - \log_2 Q, & \sigma^2 > \frac{Q^2}{2e} \\ \frac{e}{Q^2 \ln 2}\,\sigma^2, & \sigma^2 \leq \frac{Q^2}{2e}\,. \end{cases} \tag{6}$$

Note that σ^2 will be larger than $Q^2/(2e)$ for small values of Q (low distortion), and smaller for large values of Q (high distortion). The approximation in (6) for low distortion is obtained, like (3), by relating the entropy of $\hat{p}_Q(d)$ to the differential entropy of a laplacian, and for high distortion by choosing the linear function of σ^2 that makes the derivative of the formula continuous at $\sigma^2 = Q^2/(2e)$. As our estimate of σ^2 we choose

$$\sigma^2 = \frac{1}{P}\sum_{i=1}^{N} E_i, \tag{7}$$

where E_i is the expected value of the energy of the difference between the block P_i and the block in the previous frame at the location indicated by $\hat{V}_i$, the quantized version of the ith motion vector V_i. Not surprisingly, it can be shown that the expected difference energy E_i is a function of the accuracies $(\Delta_{x,i}, \Delta_{y,i})$ with which V_i is encoded. In Sections 6.1 and 6.2 of the Appendix, we show that for high motion vector accuracies (e.g., $\Delta_{x,i}, \Delta_{y,i} \leq \Delta_T$), sufficiently large and textured blocks, and in the presence of interframe noise, E_i can be approximated by

$$E_i \approx A_i\Delta_{x,i}^2 + B_i\Delta_{y,i}^2 + C_i. \tag{8}$$

where A_i, B_i and C_i are parameters that depend on P_i and the variance of the interframe noise.

The encoder estimates A_i, B_i and C_i as follows. Let $\mathcal{E}(P_i, V)$ denote the energy of the difference between P_i and the block of the previous frame corresponding to displacement V. For each $(\Delta_{x,i}, \Delta_{y,i})$ in the set of pairs of potential accuracies $\{(\Delta_T, \Delta_T), (\Delta_T, \Delta_T/2), (\Delta_T, \Delta_T/4), ..., (\Delta_T, \Delta_0), ..., (\Delta_T/2, \Delta_T), ..., (\Delta_0, \Delta_0)\}$, the encoder computes

$$\mathcal{E}(P_i, V_i + (j\Delta_0, k\Delta_0)),\ j = \frac{-\Delta_{x,i}}{2\Delta_0}, ..., \frac{\Delta_{x,i}}{2\Delta_0},\ k = \frac{-\Delta_{y,i}}{2\Delta_0}, ..., \frac{\Delta_{y,i}}{2\Delta_0}, \tag{9}$$

and their average, denoted $\bar{\mathcal{E}}_i(\Delta_{x,i}, \Delta_{y,i})$. Then the estimate for C_i is

$$C_i = \frac{1}{3}\left(\bar{\mathcal{E}}_i(\Delta_0, \Delta_0) + \bar{\mathcal{E}}_i(2\Delta_0, \Delta_0) + \bar{\mathcal{E}}_i(\Delta_0, 2\Delta_0)\right), \tag{10}$$

and the estimates for A_i and B_i are found by a least-squares fit of the $\bar{\mathcal{E}}_i(\Delta_{x,i}, \Delta_{y,i})$'s by (8) with C_i as given in (10). Observe that estimating $\{A_i, B_i, C_i\}$ is computationally expensive, but could be combined with the ideal motion estimation procedure to reduce its complexity.

Combining (5), (6), (7) and (8) yields the following approximation to the total rate for encoding the current frame

$$\bar{R} \quad \approx \quad \begin{cases} \frac{1}{2}\log_2 \frac{2e^2}{P}\sum_{i=1}^{N}\left(A_i\Delta_{x,i}^2 + B_i\Delta_{y,i}^2 + C_i\right) - \log_2 Q + \frac{1}{P}\sum_{i=1}^{N}\left(2h(\mathcal{N}) - \log_2 \Delta_{x,i}\Delta_{y,i}\right), & \bar{\sigma}^2 > \frac{Q^2}{2e} \\ \frac{e}{Q^2 \ln 2P}\sum_{i=1}^{N}\left(A_i\Delta_{x,i}^2 + B_i\Delta_{y,i}^2 + C_i\right) + \frac{1}{P}\sum_{i=1}^{N}\left(2h(\mathcal{N}) - \log_2 \Delta_{x,i}\Delta_{y,i}\right), & \bar{\sigma}^2 \leq \frac{Q^2}{2e} \end{cases} \tag{11}$$

where

$$\bar{\sigma}^2 = \frac{1}{P}\sum_{i=1}^{N}\left(A_i\Delta_{x,i}^2 + B_i\Delta_{y,i}^2 + C_i\right) \triangleq \bar{\sigma}^2_{\{\Delta_{x,i},\Delta_{y,i}\}} \tag{12}$$

3.1 Modes of operation

We have considered three different modes of operation for encoding the motion vectors for the current frame:

Mode 1: All motion vectors are encoded with the same accuracy, as in classical video coders.[1-11] Hence, $\Delta_{x,1} = \Delta_{y,1} = \ldots = \Delta_{x,N} = \Delta_{y,N} \triangleq \Delta$.

Mode 2: Different motion vectors may be encoded with different accuracies, but for a given block $\Delta_{x,i} = \Delta_{y,i} \triangleq \Delta_i$.

Mode 3: Different motion vectors may be encoded with different accuracies, and $\Delta_{x,i}$ can differ from $\Delta_{y,i}$.

For Modes 1 and 2, the set of pairs of potential accuracies is restricted to $\{(\Delta_T, \Delta_T), (\Delta_T/2, \Delta_T/2), \ldots, (\Delta_0, \Delta_0)\}$, $C_i = \bar{\mathcal{E}}(\Delta_0, \Delta_0)$, and the least-squares fit for A_i and B_i becomes one dimensional, which is much less complex. We minimized the total frame rate $\bar{R}$ in (11) for each of the three modes of operation, and derived the optimal motion vector accuracies. The results are:

Optimized Mode 1:

$$\Delta^* = \begin{cases} \Delta_1^* \triangleq L_1 \left(\frac{\mu}{\frac{1}{N}\sum_{i=1}^{N} A_i + B_i} \right)^{\frac{1}{2}}, & \bar{\sigma}^2_{\{\Delta_1^*, \Delta_1^*\}} > \frac{Q^2}{2e} \\ \Delta_2^* \triangleq L_2 \left(\frac{D}{\frac{1}{N}\sum_{i=1}^{N} A_i + B_i} \right)^{\frac{1}{2}}, & \text{otherwise,} \end{cases} \tag{13}$$

where $L_1 = \sqrt{2/(1-2N/P)}$ and $L_2 = \sqrt{12/e}$ are constants, $D = Q^2/12$ is the approximate distortion of the encoded frame, and $\mu = \frac{1}{P}\sum_{i=1}^{N} C_i$.

Optimized Mode 2:

$$\Delta_i^* = \begin{cases} \Delta_{i,1}^* \triangleq L_1 \left(\frac{\mu}{A_i + B_i} \right)^{\frac{1}{2}}, & \bar{\sigma}^2_{\{\Delta_{i,1}^*, \Delta_{i,1}^*\}} > \frac{Q^2}{2e} \\ \Delta_{i,2}^* \triangleq L_2 \left(\frac{D}{A_i + B_i} \right)^{\frac{1}{2}}, & \text{otherwise.} \end{cases} \tag{14}$$

Optimized Mode 3:

$$\Delta_{x,i}^* = \begin{cases} \Delta_{x,i,1}^* \triangleq L_1 \left(\frac{\mu}{2A_i} \right)^{\frac{1}{2}}, & \bar{\sigma}^2_{\{\Delta_{x,i,1}^*, \Delta_{y,i,1}^*\}} > \frac{Q^2}{2e} \\ \Delta_{x,i,2}^* \triangleq L_2 \left(\frac{D}{2A_i} \right)^{\frac{1}{2}}, & \text{otherwise.} \end{cases}$$

$$\Delta_{y,i}^* = \begin{cases} \Delta_{y,i,1}^* \triangleq L_1 \left(\frac{\mu}{2B_i} \right)^{\frac{1}{2}}, & \bar{\sigma}^2_{\{\Delta_{x,i,1}^*, \Delta_{y,i,1}^*\}} > \frac{Q^2}{2e} \\ \Delta_{y,i,2}^* \triangleq L_2 \left(\frac{D}{2B_i} \right)^{\frac{1}{2}}, & \text{otherwise.} \end{cases} \tag{15}$$

The formulas above show how the optimal motion accuracies depend on the parameters $\{A_i, B_i, C_i\}$ and Q, and apply when the motion accuracies are smaller than or equal to Δ_T. We note that:

- $\{A_i, B_i\}$ tend to be large for high textured blocks and small for low textured blocks. (This is made more clear later in (19) and (20)). And hence our equations indicate that motion vectors of higher textured blocks must be encoded more accurately, because A_i and B_i appear in the denominator of the Δ^*'s.
- The parameter μ is the estimated variance $\bar{\sigma}^2$ of the difference frame in the ideal case where all the motion vectors are encoded with infinite accuracy (i.e., the Δ's are zero). We refer to μ as the *noise floor* of the frame because it cannot be reduced by more accurate motion compensation. Note that μ would be zero if the motion-compensated prediction could be made identical to the current frame, which is not possible in real scenes because of camera noise, light changes, occlusions, distortion, etc. Our equations show that for larger values of μ lower motion vector accuracies are needed, because μ appears in the numerator.
- The effect of D is similar to that of μ, but for large Q, and our equations indicate that lower accuracies are necessary at higher levels of compression.
- Finally, note that the $\{A_i, B_i\}$ values increase with block size, but so do those of $\{C_i\}$ and hence the motion vectors of larger blocks must be encoded more accurately only if the increase in block texture is superior to the increase in noise floor. In fact, larger blocks may contain objects moving differently and hence the noise floor will likely be superior.

3.2 Generalization to other block-based video coders

Many other block-based video coders use difference frame encoders that are more sophisticated than the uniform scalar quantization and entropy coding of "Coder 1". For example, MPEG[2] uses a block-by-block DCT, which gives better rate-distortion performance and also exploits the response of the human visual system. The rates of those more advanced coders are expected to be lower than those of "Coder 1" by approximately a constant. However, equations (13), (14) and (15) can still be used to find the optimal motion vector accuracies because the minimization procedure is not affected by a constant difference. To confirm this, we implemented a second coder that uses a block-by-block DCT coder for the difference frame similar to MPEG's.[2] We refer to this coder as "Coder 2".

4 EXPERIMENTAL RESULTS

We present results of the performance of our two video coders on the well-known, real video sequence "caltrain" (a scene with a moving train and a calendar), for different levels of distortion and different modes of operation. We used block-matching on 8x8 blocks with $\Delta_0 = 1/64$ and the minimum absolute difference criteria[1] to approximately compute the ideal motion vectors. Where needed, bilinear interpolation was used to find the intensity values at subpixel locations. Also, the motion vector accuracies predicted by (13), (14), (15), were rounded to the nearest $\Delta_r = 2^{-r}$ (r is an integer), and were not allowed to be larger than Δ_T.

In "Coder 1", the first frame was encoded by a simple uniform scalar quantizer with quantization step Q and its rate is not included in the results. In Figure 2, the continuous lines show the total empirical rate R of "Coder 1" for encoding 4 frames of "caltrain" using Mode 1 with different values of Δ. Each line is computed for a different value of Q. From top to bottom, the continuous lines correspond to Q= 1 (lossless case), and Q= 5, 15, 25 (lossy case). (Q=25 corresponds to a peak SNR of approximately 31 dB.) The "o" signs on the Δ axis show our prediction (13) for the values Δ^* where the minima of the R lines occur. From right to left, the "o" signs correspond to the R lines for Q= 1, 5, 15, 25. The "x" signs on each R line indicate the total empirical rate that results when Δ is chosen to be the "o" value rounded to the nearest Δ_r (for Mode 1). The "+" and "*" signs plot (Δ^*_{avg}, R^*) for the optimized Modes 2 and 3, respectively, where R^* is the total empirical rate and Δ^*_{avg} is the average of the optimal motion accuracies. From top to bottom, the "+" and "*" signs correspond to cases Q= 1, 5, 15, 25.

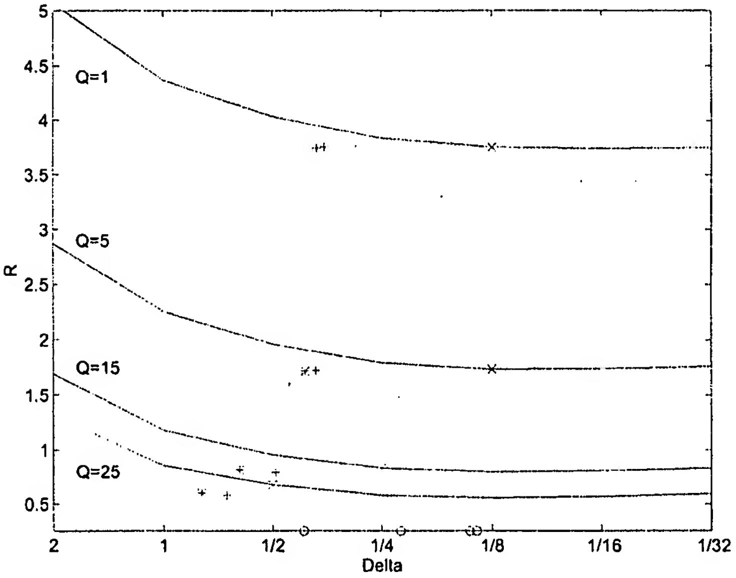

Figure 2: Rate of "Coder 1" vs. Δ for different modes at different levels of distortion

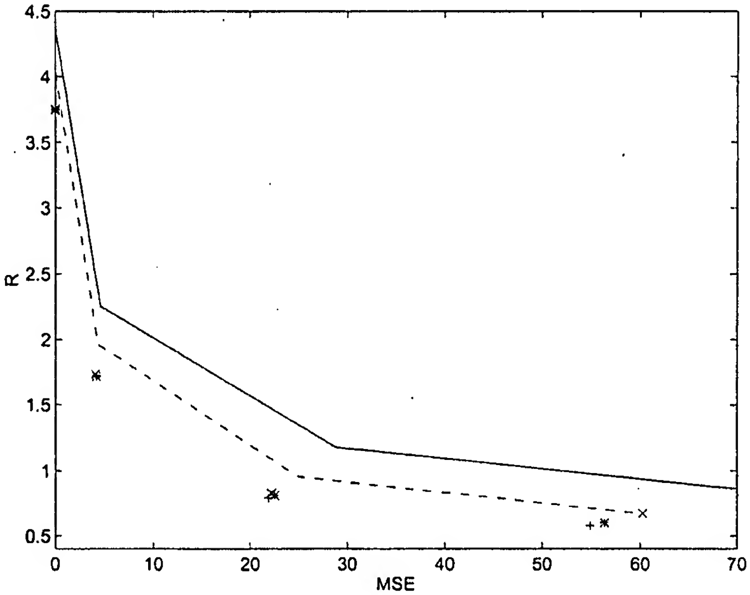

Figure 3: Rate of "Coder 1" vs. MSE distortion for typical and optimized modes

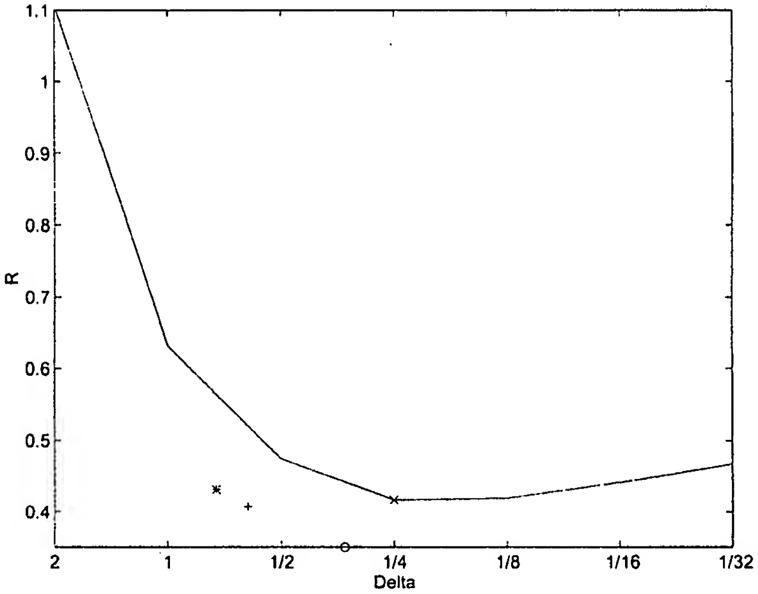

Figure 4: Rate of "Coder 2" vs. Δ for different modes at 33 dB ($Q = 20$)

Figure 3 shows the averaged rate-distortion curves for different modes of "Coder 1". Distortion is given as the mean squared error between the original and compressed frames. The continuous line is the rate-distortion curve for Mode 1 with motion vector accuracy Δ=1, and the dashed line is the rate-distortion curve for Mode 1 with Δ=1/2. The "×" signs indicate the rate-distortion values for optimized Mode 1, and the "+" and "*" signs indicate the rate-distortion values for optimized Modes 2, and 3, respectively.

Figure 4 is like Figure 2, but for encoding 30 frames of "caltrain" with "Coder 2" at a peak SNR of approximately 33 dB (accordingly, we used $Q = 20$ in our formulas). Every fifth frame was intracoded by a JPEG coder at the same distortion and its rate is not included in the results.

Finally, Figure 5 shows histograms of the $\Delta_{x,i}$ values for optimized Mode 3 in a frame of "caltrain" when Q=1 (top) and Q=25 (bottom).

5 CONCLUSIONS

Observe that our equations and results show that in our two classical block-based video coders:

- Lossless and lossy coding are relatively insensitive to the number of motion bits provided that Δ is small enough (because the curves of rate vs. Δ in Figures 2 and 4 are flat in the vicinity of their minima).
- The rate for optimized Mode 1 is as low as that of optimized Modes 2 and 3, and hence considering different accuracies for different motion vectors did not help much. (This might not always be the case.)
- Optimized Modes 1, 2, and 3 are consistently superior to Mode 1 with $\Delta = 1$ and $\Delta = 1/2$ (the typical motion vector accuracy choices of most video coding schemes). The optimized Modes give significant savings

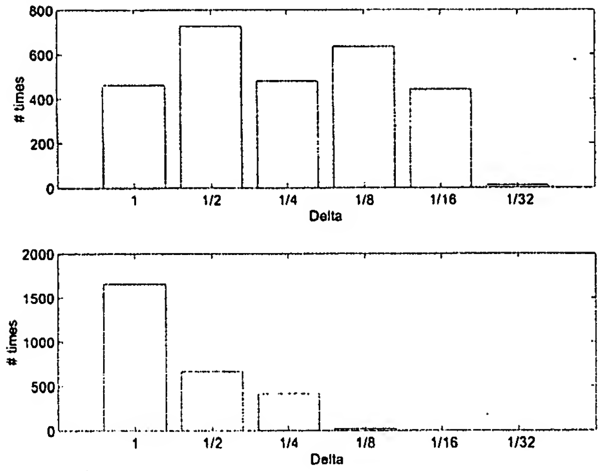

Figure 5: Histograms of the Δ values for "caltrain". *Top:* $Q = 1$, *bottom:* $Q = 25$

when Q takes values 1, 5, 15 (e.g., up to 0.7 bits/pixel respect to Δ=1), and optimized Modes 2 and 3 also save significantly for Q=25 (e.g., up to 0.12 bits/pixel or 17 per cent respect to Δ=1/2). The rate savings of the optimized modes are expected to further improve in scenes with higher texture content.

- Optimized Mode 2 is slightly superior to optimized Mode 3 for large Q, probably because the two-dimensional least-squares fit by (8) for Mode 3 is more sensitive to distortion than the one-dimensional least-squares fit for Mode 2.
- In general, the results obtained with "Coder 1" and "Coder 2" suggest that our predictions and assumptions are quite accurate. For example, (13) predicts fairly well the value Δ^* where the minimum of the rate for Mode 1 occurs, and the optimized Modes 2 and 3 are almost as good or better than Mode 1 for very small values of Δ. On the other hand, for Q=25 in "Coder 1" the prediction (13) and the optimized Mode 1 did not work as well, probably because our assumptions are less accurate for very large distortions.

Our formulas show that high motion vector accuracies are required in scenes with high texture content, and lower accuracies are needed in scenes where the prediction is corrupted by camera noise, occlusions, or other phenomena, or if the level of compression is high. Also, larger blocks will require higher motion vector accuracies if the texture increase in the blocks is superior to the increase of the noise floor. Some of these relationships have previously been observed empirically and theoretically, and some heuristic conclusions on the best motion vector accuracies have been taken and used in the design of video coders.[5,8,9,11] For example, Girod chooses to encode the motion vectors with either 1/2 pixel or 1/4 pixel accuracy.[8,9] Those may be good choices for a variety of scenes including "caltrain", but our equations indicate that lower accuracies will likely be a better choice for low textured frames encoded at large distortion, and higher accuracies will be better for high textured images at low distortion.

Our work can easily be applied to other coding schemes. For example, an object-based video coder using variable block-size motion compensation could use (13), (14) and (15) to find the optimized motion vector accuracies and reduce the bit rate. Also, our techniques have potential applications to other aspects of motion-compensated video coding. For example, when coding at low rates a variation of our procedure could be used to control the resolution of a motion estimation algorithm on a block-by-block basis in order to save a great number of operations, as well as to reduce the total bit rate. To see this, note from Figure 5 that for Q=25 almost all motion vectors are encoded with $\Delta = 1$, $1/2$, and $1/4$ accuracy, and from Figure 2 the total rate for optimized Mode 3 is nearly as low as that of Mode 1 used with much higher motion vector accuracies.

6 APPENDIX

We want to find an expression for the expected value E_i of the energy in the ith block of the difference frame, as a function of the accuracies $(\Delta_{x,i}, \Delta_{y,i})$ with which the ith motion vector is encoded. To do this, in Section 6.1 we model the energy of the difference between a continuous block and that block displaced by a small offset, in terms of the offset. (Only the major steps of this tedious analysis are indicated.) In Section 6.2, we make use of such an analysis to find a realistic model for $E_i(\Delta_{x,i}, \Delta_{y,i})$.

6.1 The effect of small displacements on difference energy

The block energy $\tilde{\mathcal{E}}$ of the difference between a rectangular block $\mathcal{P}$ in a continuous image and the same block displaced by a small offset (u_x, u_y) is

$$\tilde{\mathcal{E}}(u_x, u_y) = \int_{x=0}^{T_x} \int_{y=0}^{T_y} (\mathcal{P}(x,y) - \mathcal{P}(x+u_x, y+u_y))^2 \, dx \, dy, \tag{16}$$

where $\mathcal{P}(x,y)$ is the intensity value of the block at the continuous (x,y) coordinates, and T_x and T_y are the side lengths of the block along x and y, respectively. Our model for the intensity values of $\mathcal{P}$ is a sum of sinusoids:

$$\mathcal{P}(x,y) = \left(K_0 + \sum_{m=1}^{M} K_m \sin(\omega_{x,m} x + \omega_{y,m} y + \phi_m) \right) b_{T_x,T_y}(x,y), \tag{17}$$

where the K_m's, $\omega_{x,m}$'s, $\omega_{y,m}$'s, and ϕ_m's represent the amplitudes, frequencies (radians/sec) and phases of the sinusoids, and b_{T_x,T_y} is the two-dimensional step function of the rectangular base for the block:

$$b_{T_x,T_y}(x,y) = \begin{cases} 1 & 0 \le x \le T_x, \; 0 \le y \le T_y \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

Combining (16), (17) and (18), assuming that M is greater than or equal to 1, that u_x and u_y are small compared to T_x and T_y, and that the difference between any two $\omega_{x,m}$'s and $\omega_{y,m}$'s is larger than $2/T_x$ and $2/T_y$, respectively,† and using simple but tedious manipulations, we find

$$\begin{aligned} \tilde{\mathcal{E}}(u_x, u_y) \;&\approx\; 2K_0^2 \, (T_x T_y - (-\,|u_x| + T_x)(-\,|u_y| + T_y)) \\ &\quad - \sum_{m=1}^{M} K_m^2 \, (T_x T_y - \cos(\omega_{x,m} u_x + \omega_{y,m} u_y)(-\,|u_x| + T_x)(-\,|u_y| + T_y)), \end{aligned} \tag{19}$$

Using a Taylor expansion for the cosine, and neglecting small terms, we obtain

$$\tilde{\mathcal{E}}(u_x, u_y) \;\approx\; \left(T_x T_y \sum_{m=1}^{M} \frac{K_m^2 w_{x,m}^2}{2} \right) u_x^2 + \left(T_x T_y \sum_{m=1}^{M} \frac{K_m^2 w_{y,m}^2}{2} \right) u_y^2. \tag{20}$$

Therefore, the block energy $\tilde{\mathcal{E}}$ of the difference between a block $\mathcal{P}$ in a continuous image and the same block displaced by a small offset (u_x, u_y) follows approximately a quadratic formula:

$$\tilde{\mathcal{E}}(u_x, u_y) \approx \mathcal{A}\, u_x^2 + \mathcal{B}\, u_y^2, \tag{21}$$

where $\mathcal{A}$ and $\mathcal{B}$ are the two quadratic parameters in (20), which depend on the size $T_x T_y$ and the texture parameters $\{K_m, \omega_{x,m}, \omega_{y,m}\}$ of the block $\mathcal{P}$.

† A new preliminary analysis indicates that this assumption can be relaxed.

6.2 The quadratic model for the expected difference energy E_i

Here let $\mathcal{P}_i$ be the underlying continuous version of the block P_i in the current frame to be encoded, and let the continuous block $\mathcal{P}_i'$ be the ideal best match for $\mathcal{P}_i$ in the decoded previous frame. We can write

$$\mathcal{P}_i'(x,y) = \mathcal{P}_i(x,y) + N(x,y), \tag{22}$$

where $N(x,y)$ models the differences between $\mathcal{P}_i$ and $\mathcal{P}_i'$ due to illumination changes between frames, camera noise, distortion, etc. Let P_i'' be the prediction for P_i used by a practical encoder with a motion estimation/quantization procedure of $(\Delta_{x,i}, \Delta_{y,i})$ accuracy, and let $\mathcal{P}_i''$ be the continuous version of P_i''. Then, we model

$$\mathcal{P}_i''(x,y) = \mathcal{P}_i'(x + u_x, y + u_y), \tag{23}$$

where the motion vector errors u_x and u_y are values in $(-\Delta_{x,i}/2, \Delta_{x,i}/2)$ and $(-\Delta_{y,i}/2, \Delta_{y,i}/2)$, respectively. The energy in the ith block of the difference frame is

$$\mathcal{E}_i = \sum_{x_p=0}^{T} \sum_{y_p=0}^{T} \left(P_i(x_p, y_p) - P_i''(x_p, y_p)\right)^2, \tag{24}$$

where $P_i(x_p, y_p)$ and $P_i''(x_p, y_p)$ are the pixel values of blocks P_i and P_i'', respectively, at the discrete block coordinates (x_p, y_p), and T is the side length of the block. We approximate $\mathcal{E}_i$ (24) by

$$\mathcal{E}_i \approx \int_{x=0}^{T} \int_{y=0}^{T} \left(\mathcal{P}_i(x,y) - \mathcal{P}_i''(x,y)\right)^2 \, dx\, dy. \tag{25}$$

Combining (21), (22), (23), and (25) produces the following approximation for the difference energy $\mathcal{E}_i$ associated to P_i

$$\begin{aligned} \mathcal{E}_i &\approx \int_{x=0}^{T} \int_{y=0}^{T} \left(\mathcal{P}_i(x,y) - (\mathcal{P}_i(x+u_x, y+u_y) + N(x+u_x, y+u_y))\right)^2 dx\, dy \\ &= \int_{x=0}^{T} \int_{y=0}^{T} \left(\mathcal{P}_i(x,y) - \mathcal{P}_i(x+u_x, y+u_y)\right)^2 dx\, dy + C_i(u_x, u_y) \\ &\approx \mathcal{A}_i\, u_x^2 + \mathcal{B}_i\, u_y^2 + C_i(u_x, u_y), \end{aligned} \tag{26}$$

where $\mathcal{A}_i$ and $\mathcal{B}_i$ are the quadratic parameters for $\mathcal{P}_i$, and

$$\begin{aligned} C_i(u_x, u_y) &= \int_{x=0}^{T} \int_{y=0}^{T} N(x+u_x, y+u_y)^2 \, dx\, dy \\ &\quad -2 \int_{x=0}^{T} \int_{y=0}^{T} N(x+u_x, y+u_y) \left(\mathcal{P}_i(x,y) - \mathcal{P}_i(x+u_x, y+u_y)\right) dx\, dy. \end{aligned} \tag{27}$$

Finally, we model $N(x,y)$ as a realization of zero-mean stationary noise with variance σ_N^2, and u_x and u_y as the outcomes of two random variables (independent of the $N(x,y)$'s) uniformly distributed in $(-\Delta_{x,i}/2, \Delta_{x,i}/2)$ and $(-\Delta_{y,i}/2, \Delta_{y,i}/2)$, respectively. Combining (26) and (27), and computing the expected value, we obtain the desired model for the expected difference energy E_i

$$E_i = E[\mathcal{E}_i] \approx \mathcal{A}_i \frac{\Delta_{x,i}^2}{12} + \mathcal{B}_i \frac{\Delta_{y,i}^2}{12} + C_i = A_i \Delta_{x,i}^2 + B_i \Delta_{y,i}^2 + C_i, \tag{28}$$

with $C_i = \sigma_N^2 T^2$, $A_i = \mathcal{A}_i/12$, and $B_i = \mathcal{B}_i/12$. The approximation in (28) improves for smaller values of u_x and u_y, and for larger values of T. In practice, this quadratic model for E_i appears to work well for a wide variety of blocks in real images if u_x and u_y are smaller than or equal to 2, and T is larger than or equal to 8.

Remark: The quadratic model (26) appears to match the empirical results obtained by Vandendorpe *et al.*.[10] (Note that $C_i(u_x, u_y)$ is approximately constant for small values of u_x and u_y.)

7 REFERENCES

[1] A. Netravali and B. Haskell, *Digital pictures. Representation and Compression*, Plenum Press, 1988.

[2] D. Le Gall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, Vol. 34, pp. 47-58, Apr. 91.

[3] M. Liou, "Overview of the px64 kbit/s video coding standard," *Commun. ACM*, Vol. 34, pp. 59-63, Apr. 91.

[4] S. Eckart and C. Fogg, "ISO/IEC MPEG-2 software video codec," *Proc. SPIE Dig. Video Compr.: Alg. and Tech. 1995*, pp. 100-109, San Jose, Feb. 95.

[5] M. Hötter, "Optimization and efficiency of an object-oriented analysis-synthesis coder," *IEEE Trans. Cir. and Syst. Video Tech.*, Vol. 4, pp. 181-194, Apr. 94.

[6] P. Guillotel and C. Chevance, "Comparison of motion vector coding techniques," *Proc. SPIE Visual Commun. and Image Proc.*, pp. 1594-1604, Chicago, Sept. 94.

[7] B. Girod, "The efficiency of motion-compensating prediction for hybrid coding of video sequences," *IEEE J. Sel. Areas Commun.*, Vol. 5, pp. 1140-1154, Aug. 87.

[8] B. Girod, "Motion-compensating prediction with fractional-pel accuracy," *IEEE Trans. Commun.*, Vol. 41, pp. 604-612, Apr. 93.

[9] B. Girod, "Rate-constrained motion compensation," *Proc. SPIE Visual Commun. and Image Proc.*, pp. 1026-1034, Chicago, Sept. 94.

[10] L. Vandendorpe, L. Cuvelier and B. Maison, "Statistical properties of prediction error images in motion compensated interlaced image coding", *Proc. IEEE ICIP*, Vol. 3, pp. 192-195, Arlington, Oct. 95.

[11] H. Ito and N. Farvardin, "On motion compensation of wavelet coefficients," *Proc. IEEE ICASSP*, pp. 2161-2164, Detroit, May 95.

[12] F. Moscheni, F. Dufaux and H. Nicolas, "Entropy criterion for optimal bit allocation between motion and prediction error information," *Proc. SPIE Visual Commun. and Image Proc.*, pp. 235-242, Nov. 93.

[13] J. Ribas-Corbera and D.L. Neuhoff "Optimal bit allocations for lossless video coders: motion vectors vs. difference frames," *Proc. IEEE ICIP*, Vol. 3, pp. 180-183, Arlington, Oct. 95.

[14] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.

[15] H. Zheng and S.D. Blostein "Application of the minimum description length principle to object-oriented video image compression,' *Proc. IEEE ICIP*, pp. 400-404, Austin, Oct. 94.

JOURNAL OF VISUAL COMMUNICATION AND IMAGE REPRESENTATION
Vol. 9, No. 1, March, pp. 38–50, 1998
ARTICLE NO. VC970370

Subpixel Motion Estimation for Super-Resolution Image Sequence Enhancement*

Richard R. Schultz

Department of Electrical Engineering, University of North Dakota, P.O. Box 7165, Grand Forks, North Dakota 58202-7165
E-mail: rschultz@nyquist.ee.und.nodak.edu

Li Meng

Digital Video Products, LSI Logic Corporation, 1525 McCarthy Boulevard, Milpitas, California 95035
E-mail: lmeng@lsil.com

and

Robert L. Stevenson

Laboratory for Image and Signal Analysis, Department of Electrical Engineering, University of Notre Dame, Notre Dame, Indiana 46556
E-mail: Stevenson.1@nd.edu

Received June 17, 1997; accepted December 31, 1997

Super-resolution enhancement algorithms are used to estimate a high-resolution video still (HRVS) from several low-resolution frames, provided that objects within the digital image sequence move with subpixel increments. A Bayesian multiframe enhancement algorithm is presented to compute an HRVS using the spatial information present within each frame as well as the temporal information present due to object motion between frames. However, the required subpixel-resolution motion vectors must be estimated from low-resolution and noisy video frames, resulting in an inaccurate motion field which can adversely impact the quality of the enhanced image. Several subpixel motion estimation techniques are incorporated into the Bayesian multiframe enhancement algorithm to determine their efficacy in the presence of global data transformations between frames (i.e., camera pan, rotation, tilt, and zoom) and independent object motion. Visual and quantitative comparisons of the resulting high-resolution video stills computed from two video frames and the corresponding estimated motion fields show that the eight-parameter projective motion model is appropriate for global scene changes, while block matching and Horn–Schunck optical flow estimation each have their own advantages and disadvantages when used to estimate independent object motion. © 1998 Academic Press

1. INTRODUCTION

Super-resolution enhancement techniques may be used to estimate a high-resolution still from several low-resolution video frames, provided that objects within the image sequence move with subpixel increments [1]. To properly incorporate temporal correlations into the multiframe observation model, high-quality subpixel motion vectors must be estimated between video frames [2–4]. Methods for attaining subpixel accuracy generally employ an interpolation of the image sequence frames, followed by the application of a parametric or nonparametric motion estimation scheme. The accuracy of the estimated motion fields has a direct influence on the quality of the high-resolution video still (HRVS) image.

The concept of multiframe enhancement was originally introduced by Tsai and Huang [5], in which an observation model was defined for a sequence consisting of subpixel shifts of the same scene. Stark and Oskoui formulated a projection onto convex sets (POCS) algorithm to compute an estimate from observations obtained by scanning or rotating an image with respect to the image acquisition sensor array. Tekalp *et al.* [6] first extended this POCS formulation to include sensor noise, and then Patti [7] incorporated interlaced frames and other video sampling patterns into this algorithm. Bayesian methods of image sequence frame integration were introduced by Cheeseman [8] and Schultz and Stevenson [1]. The multiframe

* This research was supported in part by the National Science Foundation Faculty Early Career Development (CAREER) Program, Grant MIP-9624849. In addition, this material is based upon work supported in part by the U.S. Army Research Office under Contract DAAH04-96-1-0449. Research was presented in part at Visual Communications and Image Processing '97, (San Jose, CA), February, 1997.

1047-3203/98 $25.00

resolution enhancement algorithm proposed by Schultz and Stevenson [1] uses a Bayesian maximum *a posteriori* (MAP) estimation technique which incorporates a discontinuity-preserving prior to integrate progressively scanned video frames. A more general video observation model was also defined for the motion-compensated scan conversion of interlaced image sequences [9]. This research extends the Bayesian HRVS algorithm by comparing several motion estimation techniques, including the estimation of the eight-parameter projective model coefficients [10], conventional block matching motion estimation [11, 12], and Horn–Schunck optical flow estimation [13], to determine the efficacy of these methods with respect to the quality of the corresponding multiframe enhanced images.

The paper is organized as follows. Section 2 describes the Bayesian multiframe resolution enhancement technique which will be used in the simulations. An overview of three motion estimation techniques is provided in Section 3. The detection and elimination of inaccurate motion vectors may help improve the quality of localized regions within a high-resolution video still, and an algorithm used for this purpose is also proposed in Section 3. Simulation results are presented for two image sequences in Section 4, including a synthetic image sequence with a known subpixel camera pan between frames and an actual image sequence containing objects which move independently. A brief conclusion and future research directions are provided in Section 5.

2. BAYESIAN MULTIFRAME RESOLUTION ENHANCEMENT

A short low-resolution digital image sequence consisting of M frames is defined as

$$\{\mathbf{y}^{(l)}\} \quad \text{for } l = k - \left\lfloor \frac{M-1}{2} \right\rfloor, \ldots, k-1, k, \\ k+1, \ldots, k + \left\lceil \frac{M-1}{2} \right\rceil, \tag{1}$$

where $\mathbf{y}^{(k)}$ is the reference frame situated at or near the center of the sequence. In this expression, $\lfloor x \rfloor$ denotes the largest integer less than or equal to x, and $\lceil x \rceil$ represents the smallest integer greater than or equal to x. The goal of single frame image enhancement techniques—most notably bilinear [14], cubic B-spline [15], and Bayesian MAP interpolation [16]—is to estimate the unknown high-resolution image $\mathbf{z}^{(k)}$ from the low-resolution reference frame $\mathbf{y}^{(k)}$. However, if objects in the short image sequence move with subpixel increments from frame-to-frame, this additional temporal information can be utilized to improve the quality of the high-resolution estimate of $\mathbf{z}^{(k)}$. In this section, a motion-compensated subsampling model is described, along with the resulting Bayesian multiframe resolution enhancement algorithm.

2.1. *Image Sequence Observation Model*

The subsampling model which maps $q \times q$ high-resolution pixels within frame k, denoted as $z_{r,s}^{(k)}$, into a single low-resolution pixel $y_{\mathbf{x}}^{(k)}$ at spatial location $\mathbf{x} = (x_1, x_2)$ is given as

$$y_{\mathbf{x}}^{(k)} = \frac{1}{q^2}\left(\sum_{r=qx_1-q+1}^{qx_1} \sum_{s=qx_2-q+1}^{qx_2} z_{r,s}^{(k)}\right) \tag{2}$$

for $x_1 = 1, \ldots, N_1$ and $x_2 = 1, \ldots, N_2$. This represents the integration of light over a single low-resolution charge-coupled device (CCD) image acquisition sensor containing $q \times q$ high-resolution sensors within its spatial area. In matrix–vector notation, this can be expressed as

$$\mathbf{y}^{(k)} = \mathbf{A}^{(k,k)}\mathbf{z}^{(k)}, \tag{3}$$

where $\mathbf{A}^{(k,k)} \in \mathbb{R}^{N_1N_2 \times q^2N_1N_2}$ is referred to as the subsampling matrix. Intuitively, each row of $\mathbf{A}^{(k,k)}$ maps a square block of $q \times q$ high-resolution samples into a single low-resolution pixel.

If the motion vectors are known exactly, the k^{th} frame after compensation should be identical to the l^{th} frame, such that

$$y_{\mathbf{x}}^{(l)} = y_{x_1-v(\mathbf{x}),x_2-h(\mathbf{x})}^{(k)}, \tag{4}$$

where the vertical and horizontal displacement vector components at location $\mathbf{x} = (x_1, x_2)$ are denoted as $v(\mathbf{x})$ and $h(\mathbf{x})$, respectively. These displacement values may be fractional to represent subpixel-resolution motion between frames. Assuming that all motion vectors between frames k and l are known exactly and that purely translational motion can be used to represent the movement of objects between the two image sequence frames, the motion-compensated subsampling model is given as

$$\begin{aligned} y_{\mathbf{x}}^{(l)} &= y_{x_1-v(\mathbf{x}),x_2-h(\mathbf{x})}^{(k)} \\ &= \frac{1}{q^2}\left(\sum_{r=qx_1-q+1}^{qx_1} \sum_{s=qx_2-q+1}^{qx_2} z_{r-qv(\mathbf{x}),s-qh(\mathbf{x})}^{(k)}\right). \end{aligned} \tag{5}$$

Therefore, the overall high-resolution to low-resolution mapping becomes

$$\mathbf{y}^{(l)} = \mathbf{A}^{(l,k)}\mathbf{z}^{(k)}. \tag{6}$$

$\mathbf{A}^{(l,k)}$ has a structure similar to $\mathbf{A}^{(k,k)}$, but with the summations over shifted sets of pixels. Since the rows of $\mathbf{A}^{(l,k)}$ which contain useful information are those for which elements of $\mathbf{y}^{(l)}$ are observed entirely from motion-compensated elements of $\mathbf{z}^{(k)}$, low-resolution pixels which are not completely observable must be detected so that the corresponding rows of $\mathbf{A}^{(l,k)}$ can be deleted in the construction of the reduced matrix $\mathbf{A}'^{(l,k)}$. Practically, the motion vectors must be estimated using one of the techniques to be presented in Section 3.

A practical video observation model useful for real image sequences is given as

$$\mathbf{y}'^{(l)} = \hat{\mathbf{A}}'^{(l,k)}\mathbf{z}^{(k)} + \mathbf{n}^{(l,k)}, \tag{7}$$

where $\hat{\mathbf{A}}'^{(l,k)}$ is an estimate of the motion-compensated subsampling matrix which both subsamples the high-resolution frame and takes into account the motion between frames, and $\mathbf{n}^{(l,k)}$ is an additive noise term representing the error in estimating $\hat{\mathbf{A}}'^{(l,k)}$ from $\mathbf{y}^{(k)}$ and $\mathbf{y}^{(l)}$. Additive noise in this case is assumed to be Gaussian-distributed. The notation $\mathbf{y}'^{(l)}$ denotes only those pixels within $\mathbf{y}^{(l)}$ which are observable entirely from motion-compensated elements of $\mathbf{z}^{(k)}$.

2.2. High-Resolution Video Stills (HRVS) Algorithm

To estimate a high-resolution video still from a low-resolution image sequence, a Bayesian MAP estimation technique [1] is employed. The Bayesian estimate computed from M low-resolution frames is given as

$$\begin{aligned}\hat{\mathbf{z}}^{(k)} = \arg\max_{\mathbf{z}^{(k)}} \log \Pr(\mathbf{z}^{(k)} | \mathbf{y}'^{(k-\lfloor(M-1)/2\rfloor)}, \ldots, \mathbf{y}'^{(k-1)}, \\ \mathbf{y}^{(k)}, \mathbf{y}'^{(k+1)}, \ldots, \mathbf{y}'^{(k+\lceil(M-1)/2\rceil)}).\end{aligned} \tag{8}$$

Applying Bayes' theorem to the posterior probability results in the following optimization problem:

$$\begin{aligned}\hat{\mathbf{z}}^{(k)} = \arg\max_{\mathbf{z}^{(k)}} \{\log \Pr(\mathbf{z}^{(k)}) + \log \Pr(\mathbf{y}'^{(k-\lfloor(M-1)/2\rfloor)}, \ldots, \\ \mathbf{y}'^{(k-1)}, \mathbf{y}^{(k)}, \mathbf{y}'^{(k+1)}, \ldots, \mathbf{y}'^{(k+\lceil(M-1)/2\rceil)} | \mathbf{z}^{(k)})\}.\end{aligned} \tag{9}$$

In this expression, $\Pr(\mathbf{z}^{(k)})$ represents the prior image density, and the conditional density models the error in estimating the motion vectors used to construct the motion compensated subsampling matrices $\hat{\mathbf{A}}^{(l,k)}$. The motion estimation error is assumed to be independent between frames, so that the joint conditional density may be written as

$$\begin{aligned}&\Pr(\mathbf{y}'^{(k-\lfloor(M-1)/2\rfloor)}, \ldots, \mathbf{y}'^{(k-1)}, \mathbf{y}^{(k)}, \mathbf{y}'^{(k+1)}, \ldots, \mathbf{y}'^{(k+\lceil(M-1)/2\rceil)} | \mathbf{z}^{(k)}) \\ &= \Pr(\mathbf{y}^{(k)}|\mathbf{z}^{(k)}) \times \prod_{\substack{l=k-\lfloor(M-1)/2\rfloor \\ l\neq k}}^{k+\lceil(M-1)/2\rceil} \Pr(\mathbf{y}'^{(l)}|\mathbf{z}^{(k)}).\end{aligned} \tag{10}$$

Since the signal $\mathbf{z}^{(k)}$ and the noise $\mathbf{n}^{(l,k)}$ are statistically uncorrelated, each of the conditional densities for $l \neq k$ is an independent noise probability density function:

$$\begin{aligned}\Pr(\mathbf{y}'^{(l)}|\mathbf{z}^{(k)}) &= \Pr(\hat{\mathbf{A}}'^{(l,k)}\mathbf{z}^{(k)} + \mathbf{n}^{(l,k)}|\mathbf{z}^{(k)}) \\ &= \Pr(\mathbf{n}^{(l,k)}|\mathbf{z}^{(k)}) \\ &= \Pr(\mathbf{n}^{(l,k)}).\end{aligned} \tag{11}$$

Thus, the optimization problem in (9) becomes

$$\begin{aligned}\hat{\mathbf{z}}^{(k)} = \arg\max_{\mathbf{z}^{(k)}} \Bigg\{ &\log \Pr(\mathbf{z}^{(k)}) + \log \Pr(\mathbf{y}^{(k)}|\mathbf{z}^{(k)}) \\ &+ \sum_{\substack{l=k-\lfloor(M-1)/2\rfloor \\ l\neq k}}^{k+\lceil(M-1)/2\rceil} \log \Pr(\mathbf{y}'^{(l)}|\mathbf{z}^{(k)}) \Bigg\}.\end{aligned} \tag{12}$$

In order to compute the MAP estimate, the prior image model $\Pr(\mathbf{z}^{(k)})$ and the conditional densities $\Pr(\mathbf{y}^{(k)}|\mathbf{z}^{(k)})$ and $\Pr(\mathbf{y}'^{(l)}|\mathbf{z}^{(k)})$ for $l \neq k$ must be defined.

The Huber–Markov random field (HMRF) model [1] is a Gibbs prior [17] which represents piece-wise smooth data, with the probability density function defined as

$$\Pr(\mathbf{z}^{(k)}) = \frac{1}{Z}\exp\left\{-\frac{1}{2\beta}\sum_{c\in\mathcal{C}} \rho_\alpha(\mathbf{d}_c^t\mathbf{z}^{(k)})\right\}. \tag{13}$$

In this expression, Z is a normalizing constant known as the partition function, β is the Gibbs prior "temperature" parameter, and c is a local group of pixels contained within the set of all image cliques $\mathcal{C}$. The quantity $\mathbf{d}_c^t\mathbf{z}^{(k)}$ is a spatial activity measure, with a small value in smooth image locations and a large value near edges. Four spatial activity measures are computed at each pixel in the high-resolution image, given by the following second-order finite differences:

$$\begin{aligned}\mathbf{d}_{\mathbf{x},1}^t\mathbf{z}^{(k)} &= z_{x_1,x_2-1}^{(k)} - 2z_{x_1,x_2}^{(k)} + z_{x_1,x_2+1}^{(k)}, \\ \mathbf{d}_{\mathbf{x},2}^t\mathbf{z}^{(k)} &= 0.5z_{x_1+1,x_2-1}^{(k)} - z_{x_1,x_2}^{(k)} + 0.5z_{x_1-1,x_2+1}^{(k)}, \\ \mathbf{d}_{\mathbf{x},3}^t\mathbf{z}^{(k)} &= z_{x_1-1,x_2}^{(k)} - 2z_{x_1,x_2}^{(k)} + z_{x_1+1,x_2}^{(k)}, \\ \mathbf{d}_{\mathbf{x},4}^t\mathbf{z}^{(k)} &= 0.5z_{x_1-1,x_2-1}^{(k)} - z_{x_1,x_2}^{(k)} + 0.5z_{x_1+1,x_2+1}^{(k)}.\end{aligned}$$

The likelihood of discontinuities is controlled by the Huber edge penalty function [1],

$$\rho_\alpha(x) = \begin{cases} x^2, & |x| \leq \alpha, \\ 2\alpha|x| - \alpha^2, & |x| > \alpha, \end{cases} \tag{14}$$

where α is a threshold parameter which separates the quadratic and linear regions. A quadratic edge penalty, $\lim_{\alpha \to \infty} \rho_\alpha(x) = x^2$, characterizes the Gauss–Markov random field prior model. The use of the Huber function with the proper value of α helps maintain discontinuities in the image estimate.

Each conditional density, $\Pr(\mathbf{y}^{(k)}|\mathbf{z}^{(k)})$ and $\Pr(\mathbf{y}'^{(l)}|\mathbf{z}^{(k)})$ for $l \neq k$, models the error in the displacement vector estimates used to construct the motion-compensated subsampling matrices $\hat{\mathbf{A}}'^{(l,k)}$. Since motion vectors are not required for frame $\mathbf{y}^{(k)}$, the density $\Pr(\mathbf{y}^{(k)}|\mathbf{z}^{(k)})$ is given as

$$\Pr(\mathbf{y}^{(k)}|\mathbf{z}^{(k)}) = \begin{cases} 1, & \mathbf{y}^{(k)} = \mathbf{A}^{(k,k)}\mathbf{z}^{(k)}, \\ 0, & \mathbf{y}^{(k)} \neq \mathbf{A}^{(k,k)}\mathbf{z}^{(k)}. \end{cases} \tag{15}$$

For other frames in the short sequence, the error in the video observation model is assumed to be Gaussian-distributed; therefore, each conditional density is given by the expression

$$\Pr(\mathbf{y}'^{(l)}|\mathbf{z}^{(k)}) = \frac{1}{(2\pi)^{(N_1N_2)/2}\sigma^{(l,k)^{N_1N_2}}} \times \exp\left\{-\frac{1}{2\sigma^{(l,k)^2}}\left\|\mathbf{y}'^{(l)} - \hat{\mathbf{A}}'^{(l,k)}\mathbf{z}^{(k)}\right\|^2\right\} \tag{16}$$

for $l = k - \lfloor(M-1)/2\rfloor, \ldots, k-1$ and $l = k+1, \ldots, k + \lceil(M-1)/2\rceil$. The error variance $\sigma^{(l,k)^2}$ for each frame is assumed to be proportional to the absolute frame index difference $|l - k|$.

By substituting Eqs. (13), (15), and (16) into (12), the Bayesian MAP estimate of the high-resolution image can be written as the constrained optimization problem

$$\hat{\mathbf{z}}^{(k)} = \arg\min_{\mathbf{z}^{(k)} \in \mathcal{X}} \left\{ \sum_{\mathbf{x}} \sum_{i=1}^{4} \rho_\alpha(\mathbf{d}^t_{\mathbf{x},i}\mathbf{z}^{(k)}) + \sum_{l=k-\lfloor(M-1)/2\rfloor}^{k+\lceil(M-1)/2\rceil} \lambda^{(l,k)}\|\mathbf{y}'^{(l)} - \hat{\mathbf{A}}'^{(l,k)}\mathbf{z}^{(k)}\|^2 \right\}, \tag{17}$$

where the constraint set is defined as

$$\mathcal{X} = \{\mathbf{z}^{(k)} : \mathbf{y}^{(k)} = \mathbf{A}^{(k,k)}\mathbf{z}^{(k)}\}. \tag{18}$$

Each frame, $\mathbf{y}'^{(l)}$ for $l \neq k$, has an associated confidence parameter, $\lambda^{(l,k)} = \beta/\sigma^{(l,k)^2}$, proportional to the confidence in the motion-compensated subsampling matrix estimate $\hat{\mathbf{A}}'^{(l,k)}$. Since the objective function in (17) is convex, the gradient projection technique [1, 18] is used to compute the unique solution $\hat{\mathbf{z}}^{(k)}$.

3. SUBPIXEL MOTION ESTIMATION TECHNIQUES

Estimating accurate subpixel motion vectors is a critically important component of modeling an image sequence for use in super-resolution enhancement algorithms. The eight-parameter projective model coordinate transformation [10], block matching [12], and Horn–Schunck optical flow motion estimation [13] are described in this section. In all cases, two successive low-resolution video frames are first up-sampled by a factor of q using either bilinear, cubic B-spline [15], or single frame Bayesian MAP interpolation [16]. The resulting up-sampled frames are denoted as $\mathbf{y}^{\uparrow(k)}$ and $\mathbf{y}^{\uparrow(l)}$; subpixel-resolution motion vectors are then estimated from these two frames. Since a single subpixel-resolution motion vector is required for each low-resolution image pixel in the image sequence observation model, the estimated motion vectors are down-sampled by averaging over $q \times q$ vector blocks.

3.1. *Eight-Parameter Projective Model*

Parametric coordinate transformation algorithms assume that objects remain stationary while the camera or the camera lens moves; this includes transformations such as pan, rotation, tilt, and zoom. If an image sequence contains a global transformation between frames, the estimated motion field can be highly accurate due to the large ratio of observed image pixels to unknown motion model parameters. The parametric model which corresponds most closely to transformations that occur in the real world is the eight-parameter projective model [10]. The model is defined as

$$x_1' = \frac{a_1x_1 + a_2x_2 + a_3}{a_7x_1 + a_8x_2 + 1}, \quad x_2' = \frac{a_4x_1 + a_5x_2 + a_6}{a_7x_1 + a_8x_2 + 1}, \tag{19}$$

where $\mathbf{x} = (x_1, x_2)$ denotes the spatial location of a pixel in frame l, $\mathbf{x}' = (x_1', x_2')$ represents the location of the transformed pixel in frame k, and $\mathbf{a} = \{a_1, \ldots, a_8\}$ denotes the eight unknown model parameters. This model can be expressed in a more compact matrix–vector form as

$$\mathbf{x}' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \frac{\mathbf{A}\mathbf{x} + \mathbf{b}}{\mathbf{c}^t\mathbf{x} + 1}. \tag{20}$$

The matrices and vectors have the following dimensions: $\mathbf{A} \in \mathbb{R}^{2\times2}$, $\mathbf{b} \in \mathbb{R}^{2\times1}$, and $\mathbf{c} \in \mathbb{R}^{2\times1}$.

The "four point" method proposed by Mann and Picard [10] is used to compute the eight-parameter projective model coefficients. By selecting four points in frame $\mathbf{y}^{\uparrow(l)}$ corresponding to four transformed points in the up-sampled reference frame $\mathbf{y}^{\uparrow(k)}$, eight nonlinear equations result for the eight unknowns. Solving these simultaneous nonlinear equations directly is not possible. To circumvent this problem, a bilinear motion model [19] approximation to the projective model is used, such that

$$x_1' = q_1x_1x_2 + q_2x_1 + q_3x_2 + q_4, \qquad (21)$$
$$x_2' = q_5x_1x_2 + q_6x_1 + q_7x_2 + q_8.$$

Now, eight linear equations result for the four selected points, resulting in a least squares solution. The projective model parameters are then related to the bilinear coefficients in an iterative algorithm, in which the up-sampled reference frame is iteratively warped into $\mathbf{y}^{\uparrow(l)}$ until the projective model parameters converge.

3.2. Block Matching Motion Estimation

The primary drawback of using parametric motion models is that they are only applicable in the presence of global motion. Simply stated, it is expected that they will fail when objects move independently. Nonparametric models do not possess this problem, so they may be used to estimate independent object motion [19]. However, due to the low ratio of observed image pixels to unknown motion model parameters, a number of the estimated motion vectors may be inaccurate.

Block matching [12] is a popular approach for estimating motion vectors from image sequences. This method assumes that the motion field is uniform over compact blocks of pixels and that the motion can be modeled as displacements of these blocks. Let a $(2p + 1) \times (2p + 1)$ pixel region in the up-sampled frame $\mathbf{y}^{\uparrow(l)}$ represent the block to be matched between frames k and l, with pixel locations contained in the set

$$R_{\mathbf{x}} = \{(r, s) | x_1 - p \leq r \leq x_1 + p; x_2 - p \leq s \leq x_2 + p\}.$$

The maximum allowable vertical and horizontal displacement of this block is d pixels, with this parameter defining the extent of the search area in the up-sampled reference frame $\mathbf{y}^{\uparrow(k)}$. Thus, candidate motion vectors are limited to the set

$$V_{\mathbf{x}} = \{(v(\mathbf{x}), h(\mathbf{x})) | -d \leq v(\mathbf{x}) \leq d, -d \leq h(\mathbf{x}) \leq d\}.$$

Using this notation, a single displacement vector can be estimated using the mean absolute difference (MAD) criterion as

$$(\hat{v}(\mathbf{x}), \hat{h}(\mathbf{x})) = \arg \min_{(v(\mathbf{x}), h(\mathbf{x})) \in V_{\mathbf{x}}} \left\{ \frac{1}{(2p+1)^2} \sum_{(r,s) \in R_{\mathbf{x}}} |y_{r,s}^{\uparrow(l)} - y_{r-v(\mathbf{x}), s-h(\mathbf{x})}^{\uparrow(k)}| \right\} \qquad (22)$$

for $x_1 = 1, \ldots, N_1$ and $x_2 = 1, \ldots, N_2$.

Block matching generally performs well if a number of spatial discontinuities are present within the block, but the technique fails to estimate vectors properly over flat image intensity regions. A large block size, say, $p = 5$, may be used to ensure that a sufficient spatial variation exists within the block.

3.3. Horn–Schunck Optical Flow Estimation

Horn–Schunck optical flow estimation [13] results in motion vector estimates which satisfy the optical flow equation with the minimum pixel-to-pixel variation in the velocity field. Assume that continuous image intensity, denoted as $E(x_1, x_2, t)$, is constant along a particular object trajectory, such that the optical flow equation is given as

$$E(x_1, x_2, t) = E(x_1 + \Delta x_1, x_2 + \Delta x_2, t + \Delta t) \quad \forall x_1, x_2, t. \qquad (23)$$

Through a first-order Taylor series expansion of the right-hand side of Eq. (23), the optical flow equation can be rewritten as

$$\varepsilon_{of}(v(\mathbf{x}), h(\mathbf{x})) = v(\mathbf{x})\hat{E}_{x_1} + h(\mathbf{x})\hat{E}_{x_2} + \hat{E}_t \approx 0, \qquad (24)$$

where the translational flow velocity vector consists of spatial derivatives; i.e.,

$$v(\mathbf{x}) = \frac{dx_1}{dt}, \quad h(\mathbf{x}) = \frac{dx_2}{dt}. \qquad (25)$$

In the discretized optical flow equation, spatial derivative estimates, $\hat{E}_{x_1}$ and $\hat{E}_{x_2}$, and the temporal derivative estimate, $\hat{E}_t$, can be calculated from the interpolated frames as finite differences [13]:

$$\hat{E}_{x_1} = \tfrac{1}{4}[y_{x_1+1,x_2}^{\uparrow(k)} - y_{x_1,x_2}^{\uparrow(k)} + y_{x_1+1,x_2+1}^{\uparrow(k)} - y_{x_1,x_2+1}^{\uparrow(k)}$$
$$+ y_{x_1+1,x_2}^{\uparrow(l)} - y_{x_1,x_2}^{\uparrow(l)} + y_{x_1+1,x_2+1}^{\uparrow(l)} - y_{x_1,x_2+1}^{\uparrow(l)}]$$
$$\hat{E}_{x_2} = \tfrac{1}{4}[y_{x_1,x_2+1}^{\uparrow(k)} - y_{x_1,x_2}^{\uparrow(k)} + y_{x_1+1,x_2+1}^{\uparrow(k)} - y_{x_1+1,x_2}^{\uparrow(k)}$$
$$+ y_{x_1,x_2+1}^{\uparrow(l)} - y_{x_1,x_2}^{\uparrow(l)} + y_{x_1+1,x_2+1}^{\uparrow(l)} - y_{x_1+1,x_2}^{\uparrow(l)}]$$
$$\hat{E}_t = \tfrac{1}{4}[y_{x_1,x_2}^{\uparrow(l)} - y_{x_1,x_2}^{\uparrow(k)} + y_{x_1+1,x_2}^{\uparrow(l)} - y_{x_1+1,x_2}^{\uparrow(k)}$$
$$+ y_{x_1,x_2+1}^{\uparrow(l)} - y_{x_1,x_2+1}^{\uparrow(k)} + y_{x_1+1,x_2+1}^{\uparrow(l)} - y_{x_1+1,x_2+1}^{\uparrow(k)}].$$

The motion field is then estimated using the following criterion:

$$(\hat{\mathbf{v}}, \hat{\mathbf{h}}) = \arg \min_{(\mathbf{v},\mathbf{h})} \int_{x_2} \int_{x_1} [\varepsilon_{of}^2(v(\mathbf{x}), h(\mathbf{x})) + \tau^2 \varepsilon_s^2(v(\mathbf{x}), h(\mathbf{x}))] \, dx_1 \, dx_2. \qquad (26)$$

The smoothness measure, $\varepsilon_s^2(v(\mathbf{x}), h(\mathbf{x}))$, is added to the optical flow constraint to ensure that the displacement vectors vary only slightly over a small neighborhood. For each point within the image, the smoothness measure is defined as

$$\varepsilon_s^2(v(\mathbf{x}), h(\mathbf{x})) = \left(\frac{\partial v(\mathbf{x})}{\partial x_1}\right)^2 + \left(\frac{\partial v(\mathbf{x})}{\partial x_2}\right)^2 + \left(\frac{\partial h(\mathbf{x})}{\partial x_1}\right)^2 + \left(\frac{\partial h(\mathbf{x})}{\partial x_2}\right)^2. \tag{27}$$

The parameter τ^2 controls the influence of the smoothness term, and thus the smoothness of the estimated optical flow field. Minimization of (26) can be performed by solving a pair of linear equations using Gauss–Seidel iterations [13]. Simulations have shown that it takes at least 2000 iterations to achieve convergence, even for a pair of relatively small video frames.

Theoretically, Horn–Schunck optical flow estimation should yield more accurate motion vectors than block matching, since the vectors are correlated throughout the motion field. However, estimating accurate gradients from image data is difficult, and for this reason $\varepsilon_{of}^2(v(\mathbf{x}), h(\mathbf{x}))$ may be a poor estimate of the true constraints. In practice, motion fields estimated by applying the Horn–Schunck technique to actual image sequence frames contain a number of errors, since they are computed by minimizing an inaccurate objective function.

3.4. *Detection and Elimination of Inaccurate Motion Estimates*

In the multiframe enhancement algorithm, an inaccurate motion vector can cause a great deal of damage to the estimated high-resolution video still, even more so than by assuming that there is no motion at that point. Therefore, the detection and elimination of inaccurate motion vectors is an important step in the enhancement algorithm to improve the overall quality of the high-resolution video still.

The displaced frame difference (DFD) at location $\mathbf{x}$ between the up-sampled frame l and the compensated image from frame k is denoted as

$$DFD_{\mathbf{x}}^{(l,k)} = |y_{\mathbf{x}}^{\uparrow(l)} - y_{x_1-\hat{v}(\mathbf{x}),x_2-\hat{h}(\mathbf{x})}^{\uparrow(k)}|. \tag{28}$$

Ideally, the DFD should be zero if the displacement vectors describe the motion exactly. However, there is always some error associated with the incorrect estimation of the motion vectors. Therefore, the DFD has a nonzero value which can serve as a criterion of how well the motion vectors have been estimated.

The mean, μ_{DFD}, and variance, σ_{DFD}^2, of the DFD are denoted as

$$\mu_{DFD} = \frac{1}{N}\sum_{\mathbf{x}} DFD_{\mathbf{x}}^{(l,k)} = \frac{1}{N}\sum_{\mathbf{x}} |y_{\mathbf{x}}^{\uparrow(l)} - y_{x_1-\hat{v}(\mathbf{x}),x_2-\hat{h}(\mathbf{x})}^{\uparrow(k)}| \tag{29}$$

$$\sigma_{DFD}^2 = \frac{1}{N-1}\sum_{\mathbf{x}} (DFD_{\mathbf{x}}^{(l,k)} - \mu_{DFD})^2, \tag{30}$$

where N is the total number of pixels under consideration. Since it is assumed that the motion vectors along the borders of each frame are not useful due to pixels entering and leaving the scene, N is given as $N_1 \times N_2$ minus the number of excluded border pixels. A motion vector at point $\mathbf{x}$ is detected as an inaccurate estimate if $DFD_{\mathbf{x}}^{(l,k)}$ exceeds a threshold T which is dependent on the signal statistics. The threshold value used in the simulations is given as

$$T = \mu_{DFD} + 2\sigma_{DFD}. \tag{31}$$

A simple change detection algorithm examines the straight frame difference (FD) between the up-sampled frames k and l,

$$FD_{\mathbf{x}}^{(l,k)} = |y_{\mathbf{x}}^{\uparrow(l)} - y_{\mathbf{x}}^{\uparrow(k)}|. \tag{32}$$

If $FD_{\mathbf{x}}^{(l,k)}$ is small, this implies that the intensity of pixel $\mathbf{x}$ in frame l is almost the same as the corresponding pixel in frame k. Therefore, the motion vector at this point will be difficult to estimate using the block matching and optical flow techniques, and it may have to be eliminated from the vector field.

In the simulations, a motion vector estimate at point $\mathbf{x}$ is considered to be accurate if the following two conditions are satisfied:

$$DFD_{\mathbf{x}}^{(l,k)} < T, \quad FD_{\mathbf{x}}^{(l,k)} > 2$$

If an inaccurate motion vector estimate is detected at location $\mathbf{x}$, $\hat{v}(\mathbf{x})$ and $\hat{h}(\mathbf{x})$ are ignored, and the pixel is considered to be observable in only one of the two video frames.

4. SIMULATIONS

In order to compare the effects of the motion estimation techniques on the quality of the multiframe enhanced images computed by the Bayesian HRVS algorithm, two image sequences were used in the simulations. The first image sequence, *Airport*, is shown in Fig. 1. *Airport* was synthetically generated from a single digital image to model a diagonal camera pan with known motion vectors. The second image sequence, *Mobile and Calendar*, is shown in Fig. 2. This sequence is composed of several objects moving

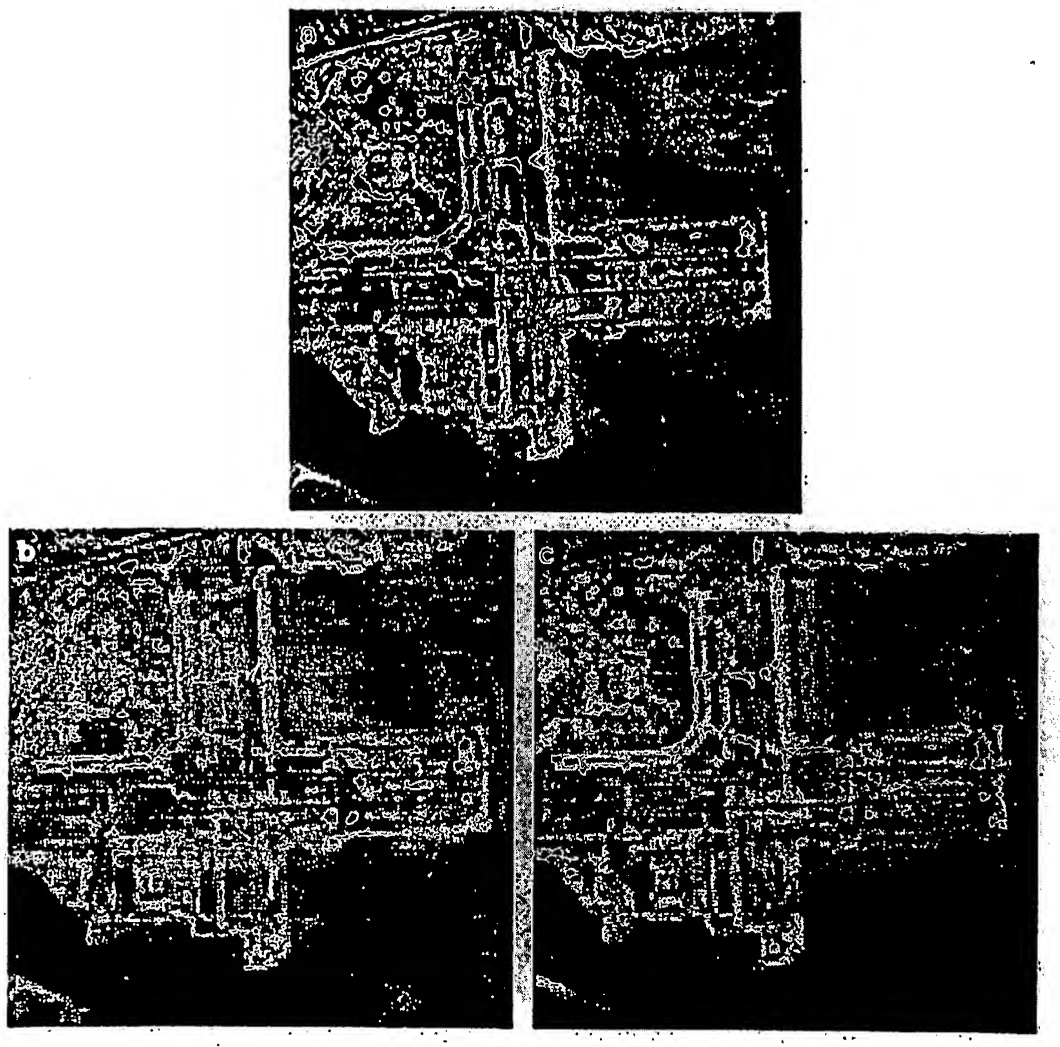

FIG. 1. *Airport* image sequence: (a) original high-resolution video frame $\mathbf{z}^{(k)}$; (b) low-resolution video frame $\mathbf{y}^{(k)}$ ($q = 2$); (c) low-resolution video frame $\mathbf{y}^{(k+1)}$ ($q = 2$).

independently. To show the effect of an estimated motion field on the quality of a high-resolution video still, only two frames within each sequence have been selected for these experiments. Thus, a high-resolution video still is computed from two low-resolution video frames and the single motion vector field that is estimated between them. It must be emphasized that we are attempting to determine empirically how the accuracy of an estimated motion field affects the quality of the high-resolution video still; with this in mind, only a single motion field (and consequently, two low-resolution video frames) can be used in the experiments. To obtain improved resolution, more frames could be added to each sequence. The reader is directed to previous research results by Schultz and Stevenson [1] to examine the visual and quantitative results that are possible when more than two frames are integrated using the Bayesian HRVS algorithm.

Both test image sequences were first subsampled by a factor of q ($q = 2$ or $q = 4$) and then interpolated back to their original dimensions so that quantitative comparisons could be made using the improved signal-to-noise ratio (SNR). The improved SNR is a quantitative measure of how much the estimated frame $\hat{\mathbf{z}}^{(k)}$ has improved over the reference frame, given as

$$\Delta_{SNR} = 10 \log_{10} \frac{\|\mathbf{z}^{(k)} - \mathbf{z}_0^{(k)}\|^2}{\|\mathbf{z}^{(k)} - \hat{\mathbf{z}}^{(k)}\|^2} \quad \text{(in dB)} \qquad (33)$$

In this expression, $\mathbf{z}_0^{(k)}$ is generated by a zero-order hold up-sampling of the reference frame $\mathbf{y}^{(k)}$, $\mathbf{z}^{(k)}$ is the original

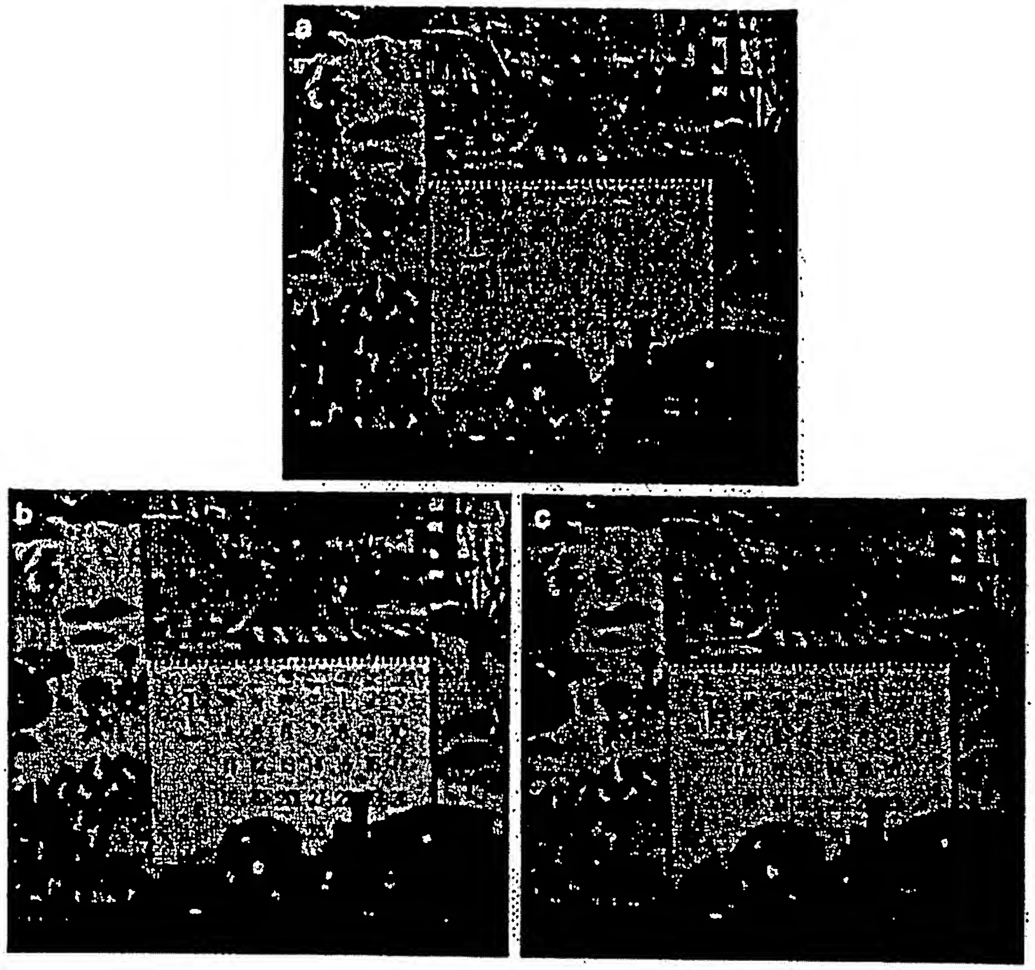

FIG. 2. *Mobile and calendar* image sequence: (a) original high-resolution video frame $\mathbf{z}^{(k)}$; (b) low-resolution video frame $\mathbf{y}^{(k)}$ ($q = 2$); (c) low-resolution video frame $\mathbf{y}^{(k+1)}$ ($q = 2$).

high-resolution image, and $\hat{\mathbf{z}}^{(k)}$ is the estimated high-resolution video still image. It should be noted that this quantitative measure can only be used in idealized test cases where $\mathbf{z}^{(k)}$ is known.

The procedure used to compute the multiframe high-resolution estimates is as follows:

1. Up-sample both low-resolution frames $\mathbf{y}^{(k)}$ and $\mathbf{y}^{(l)}$ by a factor of $q = 2$ or $q = 4$, using (a) bilinear interpolation, (b) cubic B-spline interpolation, and (c) single frame Bayesian interpolation ($M = 1$, $\alpha = 1.0$). The resulting up-sampled frames are denoted as $\mathbf{y}^{\uparrow(k)}$ and $\mathbf{y}^{\uparrow(l)}$.

2. Estimate the subpixel-resolution motion vectors between $\mathbf{y}^{\uparrow(k)}$ and $\mathbf{y}^{\uparrow(l)}$ by applying (a) eight-parameter projective model estimation ("four point" method), (b) block matching ($p = 4$, $d = 9$), and (c) Horn–Schunck optical flow estimation ($\tau^2 = 10.0$, 2000 iterations). Detect and eliminate inaccurate motion vector estimates. Finally, down-sample the displacement vectors by averaging $q \times q$ vector blocks.

3. Compute the Bayesian multiframe HRVS for each motion field using the following parameters: $M = 2$; $q = 2$ or $q = 4$; $\alpha = 1.0$; and $\lambda^{(l,k)} = 5.0/|l - k|$.

Tables 1 and 2 provide a quantitative comparison of the single frame and multiframe enhanced estimates computed from the *Airport* and *Mobile and Calendar* sequences, respectively, while Figs. 3 and 4 provide a visual comparison of several multiframe estimates computed using the three different subpixel motion estimation techniques. For all multiframe estimates shown in Figs. 3 and 4, single frame Bayesian interpolation was used to up-sample the low-resolution frames so that the subpixel motion vector fields could be estimated. Conclusions drawn from the visual and quantitative comparisons are as follows:

1. Provided that the motion is estimated correctly, val-

TABLE 1*
Comparison of *Airport* Sequence High-Resolution Video Stills
Δ_{SNR} (in dB) of Single Frame and Multiframe Resolution Enhanced Estimates

Single frame up-sampling algorithm/motion estimation technique	$q = 2$		$q = 4$	
	All	Accurate	All	Accurate
Single frame bilinear interpolation	0.93		0.59	
Single frame cubic B-spline interpolation	3.08		1.40	
Single frame Bayesian interpolation	3.44		1.67	
Bilinear interpolation/eight-parameter	6.85	7.08	2.24	2.38
Bilinear interpolation/block matching	6.12	7.18	2.30	2.34
Bilinear interpolation/Horn–Schunck	3.08	3.17	1.79	1.85
Cubic B-spline interpolation/eight-parameter	6.17	7.17	2.30	2.70
Cubic B-spline interpolation/block matching	6.11	7.14	2.34	2.38
Cubic B-spline interpolation/Horn–Schunck	2.84	3.07	1.77	1.78
Bayesian interpolation/eight-parameter	7.00	7.18	2.42	2.52
Bayesian interpolation/block matching	5.96	6.88	2.23	2.20
Bayesian interpolation/Horn–Schunck	2.82	2.96	1.57	1.70

* "All" represents the use of all estimated motion vectors, while "Accurate" denotes that the inaccurate motion vector estimates have been detected and eliminated prior to the application of the multiframe resolution enhancement algorithm. Δ_{SNR} values (in dB) represent the quantitative improvement of the high-resolution video still over the low-resolution reference frame.

ues of Δ_{SNR} should be larger for the multiframe estimates than for the single frame estimates. This is quite evident for the multiframe *Airport* estimates in Table 1 which used either the eight-parameter projective model or the block matching motion vectors and the *Mobile and Calendar* estimates in Table 2 which used either the block matching or Horn–Schunck optical flow motion vectors.

2. The quality of the subpixel-resolution motion vector

TABLE 2*
Comparison of *Mobile and Calendar* Sequence High-Resolution Video Stills
Δ_{SNR} (in dB) of Single Frame and Multiframe Resolution Enhanced Estimates

Single frame up-sampling algorithm/motion estimation technique	$q = 2$		$q = 4$	
	All	Accurate	All	Accurate
Single frame bilinear interpolation	0.27		0.35	
Single frame cubic B-spline interpolation	1.70		0.76	
Single frame Bayesian interpolation	2.41		1.12	
Bilinear interpolation/eight-parameter	−0.13	−0.04	1.01	1.03
Bilinear interpolation/block matching	2.35	2.82	1.30	1.33
Bilinear interpolation/Horn–Schunck	2.74	2.95	1.23	1.23
Cubic B-spline interpolation/eight-parameter	0.56	0.85	1.08	1.08
Cubic B-spline interpolation/block matching	2.83	3.07	1.29	1.32
Cubic B-spline interpolation/Horn–Schunck	2.54	2.84	1.16	1.18
Bayesian interpolation/eight-parameter	0.27	1.01	1.10	1.12
Bayesian interpolation/block matching	2.49	2.86	1.25	1.26
Bayesian interpolation/Horn–Schunck	2.58	2.79	1.15	1.17

* "All" represents the use of all estimated motion vectors, while "Accurate" denotes that the inaccurate motion vector estimates have been detected and eliminated prior to the application of the multiframe resolution enhancement algorithm. Δ_{SNR} values (in dB) represent the quantitative improvement of the high-resolution video still over the low-resolution reference frame.

estimates is dependent upon the accuracy of the up-sampled image data. Bilinear, cubic B-spline, and single frame Bayesian MAP interpolation ($M = 1, \alpha = 1.0$) were used to up-sample the low-resolution video frames prior to the application of a particular motion estimation method. Empirically, it was determined that both cubic B-spline and Bayesian MAP interpolation provide useful and very similar up-sampled frames, while bilinear interpolation is rather ineffective.

3. The performance of the three motion estimation algorithms varies for the test image sequences. In the case of global motion, the eight-parameter projective model can recover the motion extremely well. As expected, the algorithm performs poorly on sequences which contain objects that move independently. On the other hand, block matching and Horn–Schunck optical flow estimation perform quite well in this case, but not for global data transformations. It is expected that the Horn–Schunck algorithm should perform better than block matching for the general case of independent object motion. However, as shown empirically, this is not always the case. Estimating the spatial and temporal gradients as required by the Horn–Schunck algorithm results in an inaccurate objective function and, consequently, in many inaccurate motion vectors.

4. In most cases, the multiframe high-resolution images and their corresponding Δ_{SNR} values show that the quality of a particular HRVS is improved after inaccurate motion vectors have been detected and eliminated.

5. For the interpolation factor $q = 4$, Tables 1 and 2 show that the quality of the multiframe HRVS computed using $M = 2$ frames does not differ significantly from the single frame Bayesian estimate ($M = 1, \alpha = 1.0$). This implies that the estimated motion vectors are quite poor for $q > 2$, and that the subpixel-resolution motion estimation schemes used in this research do not perform well for practical interpolation factors. Parametric techniques can be useful for estimating higher-resolution motion fields between frames containing global motion (*e.g.*, reconnaissance and satellite image sequences). Estimating high-resolution motion fields between frames containing independent object motion is still an open research area, as block matching and Horn–Schunck estimation do not perform adequately.

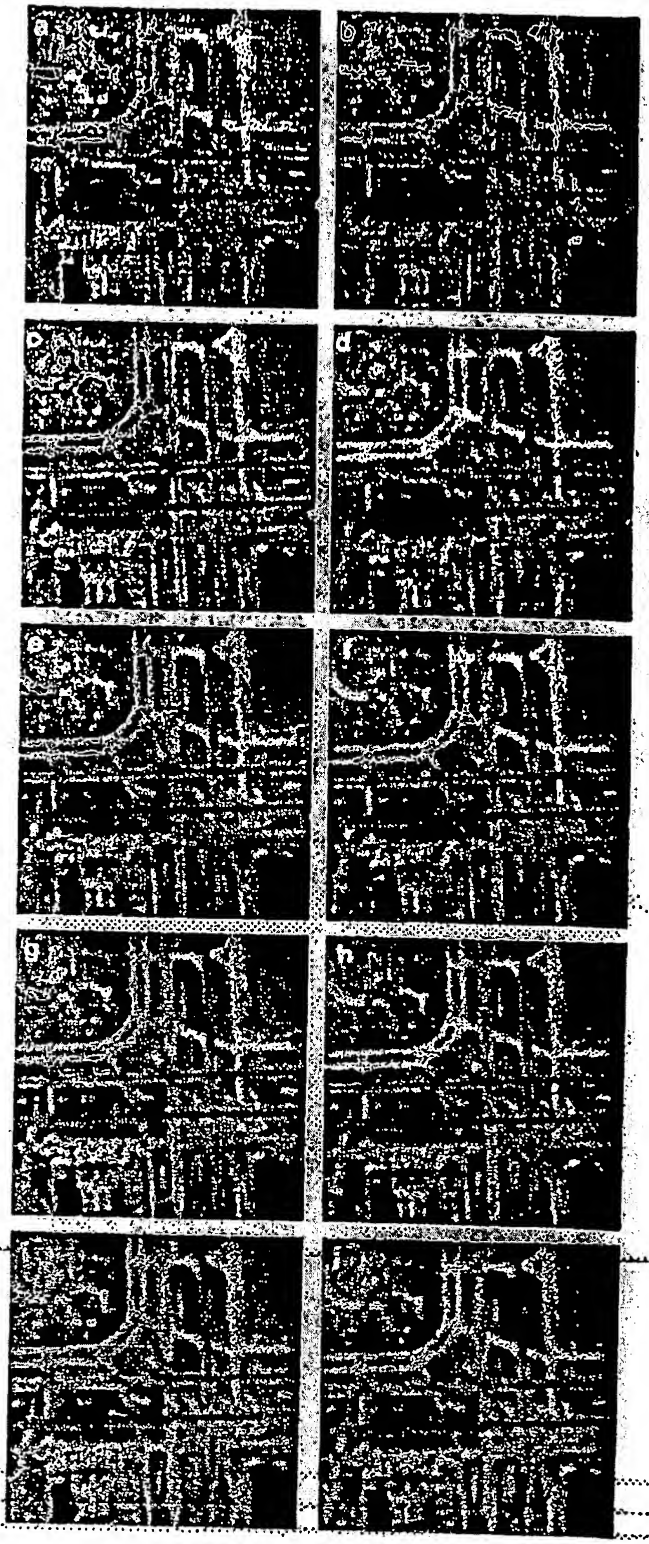

FIG. 3. Details of *Airport* sequence high-resolution video stills, $q = 2$: (a) low-res frame, $y^{(k)}$; (b) low-res frame, $y^{(k-1)}$; (c) high-res frame, $z^{(k)}$; (d) single frame Bayesian estimate, $\Delta_{SNR} = 3.44$ dB; (e) multiframe HRVS, eight-parm model, all vectors, $\Delta_{SNR} = 7.00$ dB; (f) multiframe HRVS, eight-parm model, accurate vectors, $\Delta_{SNR} = 7.18$ dB; (g) multiframe HRVS, block matching, all vectors, $\Delta_{SNR} = 5.96$ dB; (h) multiframe HRVS, block matching, accurate vectors, $\Delta_{SNR} = 6.88$ dB; (i) multiframe HRVS, Horn–Schunck, all vectors, $\Delta_{SNR} = 2.82$ dB; (j) multiframe HRVS, Horn–Schunck, accurate vectors, $\Delta_{SNR} = 2.96$ dB.

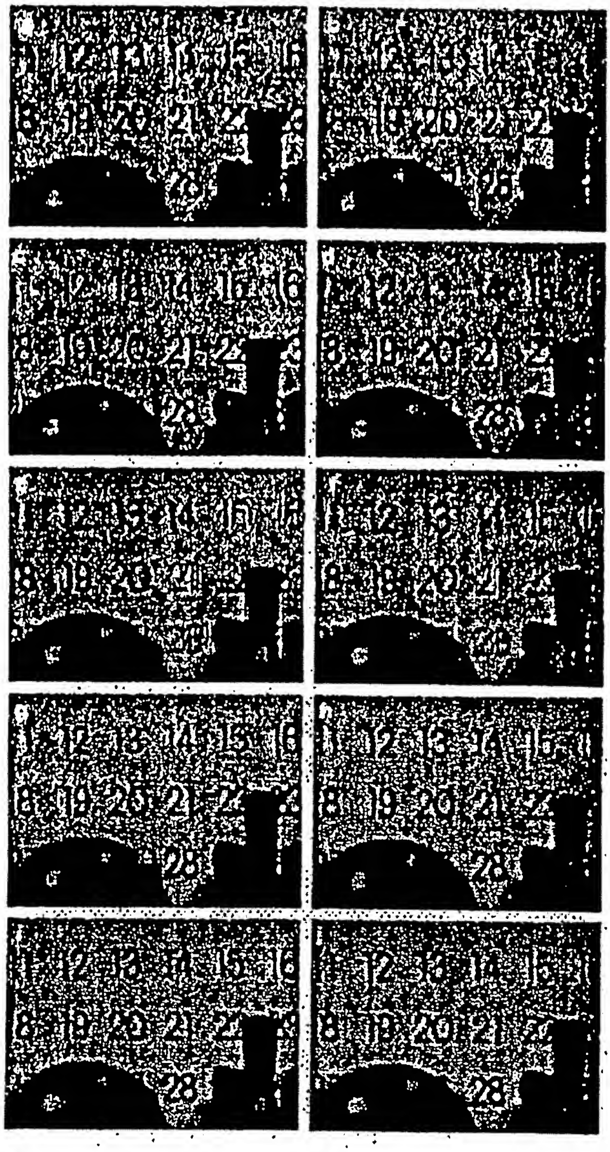

FIG. 4. Details of *Mobile and Calendar* sequence high-resolution video stills, $q = 2$: (a) low-res frame, $y^{(k)}$; (b) low-res frame, $y^{(k+1)}$; (c) high-res frame, $z^{(k)}$; (d) single frame Bayesian estimate, $\Delta_{SNR} =$ 2.41 dB; (e) multiframe HRVS, eight-parm model, all vectors, $\Delta_{SNR} = 0.27$ dB; (f) multiframe HRVS, eight-parm model, accurate vectors, $\Delta_{SNR} = 1.01$ dB; (g) multiframe HRVS, block matching, all vectors, $\Delta_{SNR} = 2.49$ dB; (h) multiframe HRVS, block matching, accurate vectors, $\Delta_{SNR} = 2.86$ dB; (i) multiframe HRVS, Horn–Schunck, all vectors, $\Delta_{SNR} = 2.58$ dB; (j) multiframe HRVS, Horn–Schunck, accurate vectors, $\Delta_{SNR} = 2.79$ dB.

5. CONCLUSION

Super-resolution image sequence enhancement methods are used to estimate a high-resolution video still from several low-resolution video frames, provided that objects within the sequence move with subpixel increments. An image sequence observation model was presented for low-resolution frames, which models the subsampling of the unknown high-resolution data and accounts for independent object motion occurring between frames. Three single frame enhancement methods were used in the experiments, including the bilinear, cubic B-spline, and single frame Bayesian interpolation techniques. The Bayesian multiframe enhancement algorithm was described to reconstruct a high-resolution video still from several low-resolution image sequence frames. In this algorithm, estimating accurate subpixel-resolution motion vectors is critically important. The eight-parameter projective parametric transformation, block matching, and Horn–Schunck optical flow estimation techniques designed for subpixel motion estimation were utilized in the computer simulations. The resulting high-resolution video stills computed using the various estimated motion fields were analyzed both visually and quantitatively to determine the most useful motion estimation technique. It was determined that the parametric motion model results in accurate multiframe video estimates when a global transformation such as a camera pan, rotation, tilt, or zoom occurs between frames. Optical flow equation-based techniques tend to perform well when objects move independently in front of a stationary camera lens. Block matching can recover the motion well when the sequence contains a large number of spatial discontinuities and few flat image regions. All motion estimation techniques are inherently imperfect, and it is easy to see the locations of inaccurate motion estimates within a high-resolution video still. An algorithm for the detection and elimination of inaccurate motion vectors was described, and simulations verified that the HRVS is improved when the inaccurate motion vectors were eliminated from the motion field.

A number of issues will be explored in future research. First, the Horn–Schunck optical flow estimation method will be modified to allow discontinuities within the motion field by incorporating the Huber edge penalty function into the smoothness measure [20]. The goal is to automatically segment the motion field into regions representing independent objects. Furthermore, the motion of each object will be studied, since a more accurate subpixel-resolution motion estimate can be obtained by averaging over the motion field region where an individual object is undergoing a transformation between frames. Secondly, an iterative motion estimation approach will be implemented for the multiframe enhancement algorithm. Explicitly, multiframe estimates will serve as the up-sampled frames, and then the subpixel-resolution motion vector estimates will be refined in an iterative fashion. An improved image sensor model is also under investigation, which incorporates motion-compensated data subsampling and a point spread function (PSF) to represent blur due to an out-of-focus camera lens or object motion.

APPENDIX: SYMBOLS

N_1	number of rows within a digital video frame
N_2	number of columns within a digital video frame
M	number of frames within a digital image sequence
$\mathbf{z}$	lexicographically-ordered digital image, represented as a vector
$\mathbf{z}^{(k)}$	frame k within a digital image sequence
$z_{\mathbf{x}}^{(k)}$	pixel at spatial location $\mathbf{x} = (x_1, x_2)$ within the k^{th} frame
$\hat{\mathbf{z}}$	an estimate of the vector $\mathbf{z}$
$\lVert\cdot\rVert$	Euclidean norm in real space
$\rho_\alpha(\cdot)$	Huber edge penalty function, dependent on parameter α
$\mathbf{d}'_{\mathbf{x},i}\mathbf{z}^{(k)}$	spatial activity measure computed at pixel $z_{\mathbf{x}}^{(k)}$
α	Huber edge penalty function threshold parameter
β	Gibbs prior temperature parameter
λ	smoothing parameter
μ	mean or average value
σ	Gaussian noise standard deviation
$\Pr(\cdot)$	probability density function
$\exp\{\cdot\}$	exponential function
$\lceil\cdot\rceil$	operator returning the least integral value greater than or equal to its argument
$\lfloor\cdot\rfloor$	operator returning the greatest integral value less than or equal to its argument
$\mathcal{X}$	constraint set
q	integer-valued interpolation factor
$\mathbf{y}'$	vector of useful observations, with reduced dimensionality
$\mathbf{A}'$	matrix of useful constraints, with reduced row dimensionality
Δ_{SNR}	improved signal-to-noise ratio, in decibels.

REFERENCES

1. R. R. Schultz and R. L. Stevenson, Extraction of high-resolution frames from video sequences, *IEEE Trans. Image Process.* 5(6), 1996, 996–1011.
2. C. A. Berenstein, L. N. Kanal, D. Lavine, and E. C. Olson, A geometric approach to subpixel registration accuracy, *Computer Vision, Graphics, and Image Processing*, 40, 1987, 334–360.
3. G. de Haan and P. W. A. C. Beizen, Sub-pixel motion estimation with 3-D recursive search block-matching, *Signal Processing: Image Communications* 6, 1994, 229–239.
4. Q. Tian and M. N. Huhns, Algorithms for subpixel registration, *Computer Vision Graphics, Image Processing* 35, 1986, 220–233.
5. R. Y. Tsai and T. S. Huang, Multiframe image restoration and registration, in *Advances in Computer Vision and Image Processing* (R. Y. Tsai and T. S. Huang, Eds.), Vol. 1, pp. 317–339, JAI Press, London, 1984.
6. A. M. Tekalp, M. K. Ozkan, and M. I. Sezan, High-resolution image reconstruction from lower-resolution image sequences and space-varying image restoration, in *Proceedings, IEEE International Conference on Acoustics, Speech, and Signal Processing, San Francisco, CA, 1992*, pp. III–169 to III–172.
7. A. J. Patti, M. I. Sezan, and A. M. Tekalp, High-resolution standards conversion of low resolution video, in *Proceedings, IEEE International Conference on Acoustics, Speech, and Signal Processing, Detroit, MI, 1995*, pp. 2197–2200.
8. P. Cheesman, B. Kanefsky, R. Kraft, J. Stutz, and R. Hanson, *Super-Resolved Surface Reconstruction from Multiple Images*, Tech. Rep. FIA-94-12, NASA Ames Research Center, Moffett Field, CA, 1994.
9. R. R. Schultz and R. L. Stevenson, Motion-compensated scan conversion of interlaced video sequences, in *Proceedings of the IS&T/SPIE Conference on Image and Video Processing IV*, vol. 2666, (San Jose, CA), pp. 107–118, 1996.
10. S. Mann and R. W. Picard, Virtual bellows: constructing high quality stills from video, in *Proceedings of the IEEE International Conference on Image Processing* (Austin, TX), pp. 363–367, 1994.
11. M. Bierling and R. Thoma, Motion compensating field interpolation using a hierarchically structured displacement estimator, *Signal Processing* 11(4), 1986, 387–404.
12. H. G. Musmann, P. Pirsch, and H.-J. Grallert, Advances in picture coding, *Proceedings of the IEEE* 73(4), 1985, 523–548.
13. B. K. P. Horn and B. G. Schunck, Determining optical flow, *Artificial Intelligence* 17, 1981, 185–203.
14. A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice–Hall, Englewood Cliffs, NJ, 1989.
15. H. H. Hou and H. C. Andrews, Cubic splines for image interpolation and digital filtering, *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26(6), 1978, 508–517.
16. R. R. Schultz and R. L. Stevenson, A Bayesian approach to image expansion for improved definition, *IEEE Transactions on Image Processing* 3(3), 1994, 233–242.
17. S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6(6), 1984, 721–741.
18. J. M. Ortega and W. C. Rheinbolt, *Iterative Solutions of Nonlinear Equations in Several Variables*. Computer Science and Applied Mathematics, New York: Academic Press, 1970.
19. A. M. Tekalp, *Digital Video Processing*. Upper Saddle River, NJ: Prentice Hall, Inc., 1995.
20. D. Shulman and J.-Y. Hervé, Regularization of discontinuous flow fields, in *Proceedings, Workshop on Visual Motion, Irvine, CA, 1989*, pp. 81–86.

RICHARD R. SCHULTZ was born on March 19, 1967, in Grafton, North Dakota. He received the B.S.E.E. degree (summa cum laude) from the University of North Dakota in 1990, and the M.S.E.E. and Ph.D. degrees from the University of Notre Dame in 1992 and 1995, respectively. He joined the faculty of the Department of Electrical Engineering at the University of North Dakota in 1995, where he is currently an Assistant Professor. In 1996, Dr. Schultz received a National Science Foundation Faculty Early Career Development (CAREER) award to integrate his image processing research and educational activities. Dr. Schultz is a member of the IEEE, SPIE, ASEE, Eta Kappa Nu, and Tau Beta Pi. His current research interests include digital signal, image, and video processing.

LI MENG was born on September 23, 1971, in Beijing, China. She received the B.S. degree in electrical engineering from Beijing University in 1995, and the M.S.E.E. degree from the University of North Dakota in 1996. Ms. Meng joined LSI Logic Corporation (Milpitas, CA) in 1997. She conducts research and development within the Digital video Products group.

ROBERT L. STEVENSON received the B.E.E. degree (summa cum laude) from the University of Delaware in 1986, and the Ph.D. in electrical engineering from Purdue University in 1990. While at Purdue, he was supported by graduate fellowships from the National Science Foundation, DuPont Corporation, Phi Kappa Phi, and Purdue University. He joined the faculty of the Department of Electrical Engineering at the University of Notre Dame in 1990, where he is currently an Associate Professor. His research interests include image/video processing, image/video compression, robust image/video communication systems, multimedia systems, ill-posed problems in computational vision, and computational issues in image processing. Dr. Stevenson currently serves as an Associate Editor for the *IEEE Transactions on Circuits and Systems for Video Technology*. He is the General Chair of the 1998 Midwest Symposium on Circuits and Systems to be held on the campus of the University of Notre Dame, and he will co-chair the 1999 Conference on Visual Communications and Image Processing.

Presented at the SPIE Conference on Applications of Digital Image Processing XXVII
Special Session on Advances in the New Emerging Standard: H.264/AVC, August, 2004

Paper No. 5558-53

The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions

Gary J. Sullivan[*], Pankaj Topiwala[†], and Ajay Luthra[‡]

[*]Microsoft Corporation, One Microsoft Way, Redmond, WA 98052
[†]FastVDO LLC, 7150 Riverwood Dr., Columbia, MD 21046
[‡]Motorola Inc., BCS, 6420 Sequence Dr., San Diego, CA 92121

ABSTRACT

H.264/MPEG-4 AVC is the latest international video coding standard. It was jointly developed by the Video Coding Experts Group (VCEG) of the ITU-T and the Moving Picture Experts Group (MPEG) of ISO/IEC. It uses state-of-the-art coding tools and provides enhanced coding efficiency for a wide range of applications including video telephony, video conferencing, TV, storage (DVD and/or hard disk based, especially high-definition DVD), streaming video, digital video authoring, digital cinema, and many others. The work on a new set of extensions to this standard has recently been completed. These extensions, known as the Fidelity Range Extensions (FRExt), provide a number of enhanced capabilities relative to the base specification as approved in the Spring of 2003. In this paper, an overview of this standard is provided, including the highlights of the capabilities of the new FRExt features. Some comparisons with the existing standards, MPEG-2 and MPEG-4 Part 2, are also provided.

Keywords: Advanced Video Coding (AVC), Digital Video Compression, H.263, H.264, JVT, MPEG, MPEG-2, MPEG-4, MPEG-4 part 10, VCEG.

1. INTRODUCTION

Since the early 1990s, when the technology was in its infancy, international video coding standards – chronologically, H.261 [1], MPEG-1 [2], MPEG-2 / H.262 [3], H.263 [4], and MPEG-4 (Part 2) [5] – have been the engines behind the commercial success of digital video compression. They have played pivotal roles in spreading the technology by providing the power of interoperability among products developed by different manufacturers, while at the same time allowing enough flexibility for ingenuity in optimizing and molding the technology to fit a given application and making the cost-performance trade-offs best suited to particular requirements. They have provided much-needed assurance to the content creators that their content will run everywhere and they do not have to create and manage multiple copies of the same content to match the products of different manufacturers. They have allowed the economy of scale to allow steep reduction in cost for the masses to be able to afford the technology. They have nurtured open interactions among experts from different companies to promote innovation and to keep pace with the implementation technology and the needs of the applications.

ITU-T H.264 / MPEG-4 (Part 10) Advanced Video Coding (commonly referred as H.264/AVC) [6] is the newest entry in the series of international video coding standards. It is currently the most powerful and state-of-the-art standard, and was developed by a Joint Video Team (JVT) consisting of experts from ITU-T's Video Coding Experts Group (VCEG) and ISO/IEC's Moving Picture Experts Group (MPEG). As has been the case with past standards, its design provides the most current balance between the coding efficiency, implementation complexity, and cost – based on state of VLSI design technology (CPU's, DSP's, ASIC's, FPGA's, etc.). In the process, a standard was created that improved coding efficiency by a factor of at least about two (on average) over MPEG-2 – the most widely used video coding standard today – while keeping the cost within an acceptable range. In July, 2004, a new amendment was added to this standard, called the Fidelity Range Extensions (FRExt, Amendment 1), which demonstrates even further coding efficiency against MPEG-2, potentially by as much as 3:1 for some key applications. In this paper, we develop an outline of the first version of the H.264/AVC standard, and provide an introduction to the newly-minted extension, which, for reasons we explain, is already receiving wide attention in the industry.

1.1. H.264/AVC History

H.264/AVC was developed over a period of about four years. The roots of this standard lie in the ITU-T's H.26L project initiated by the Video Coding Experts Group (VCEG), which issued a Call for Proposals (CfP) in early 1998 and created a first draft design for its new standard in August of 1999. In 2001, when ISO/IEC's Moving Pictures Experts Group (MPEG) had finished development of its most recent video coding standard, known as MPEG-4 Part 2, it issued a similar CfP to invite new contributions to further improve the coding efficiency beyond what was achieved on that project. VCEG chose to provide its draft design in response to MPEG's CfP and proposed joining forces to complete the work. Several other proposals were also submitted and were tested by MPEG as well. As a result of those tests, MPEG made the following conclusions that affirmed the design choices made by VCEG for H.26L:

- Motion compensated Discrete Cosine Transform (DCT) structure was superior to others, implying there was no need, at least at that stage, to make fundamental structural changes for the next generation of coding standard.
- Some video coding tools that have been excluded in the past (for MPEG-2, H.263, or MPEG-4 Part 2) due to their complexity (hence implementation cost) could be re-examined for inclusion in the next standard. The VLSI technology had advanced significantly since the development of those standards in the past and it has reduced significantly the implementation cost of those coding tools. (This was not a "blank check" for compression at all costs, as a number of compromises were still necessary for complexity reasons, but it was a recognition that some of the complexity constraints that governed past work could be re-examined.)
- To allow maximum freedom of improving the coding efficiency, the syntax of the new coding standard could not be backward compatible with prior standards.
- ITU-T's H.26L was a top-performing proposal, and most others that showed good performance in MPEG had also been based on H.26L (as it had become well-known as an advance in technology by that time).

Therefore, to allow speedy progress, ITU-T and ISO/IEC agreed to join forces together to jointly develop the next generation of video coding standard and use H.26L as the starting point. A Joint Video Team (JVT), consisting of experts from VCEG and MPEG, was formed in December, 2001, with the goal of completing the technical development of the standard by 2003. ITU-T planned to adopt the standard under the name of ITU-T H.264, and ISO/IEC planned to adopt the standard as MPEG-4 Part 10 Advanced Video Coding (AVC), in the MPEG-4 suite of standards formally designated as ISO/IEC 14496. As an unwanted byproduct, this standard gets referred to by at least six different names – H.264, H.26L, ISO/IEC 14496-10, JVT, MPEG-4 AVC and MPEG-4 Part 10. In this paper we refer it as H.264/AVC as a balance between the names used in the two organizations.

With the wide breadth of applications considered by the two organizations, the application focus for the work was correspondingly broad – from video conferencing to entertainment (broadcasting over cable, satellite, terrestrial, cable modem, DSL etc.; storage on DVDs and hard disks; video on demand etc.) to streaming video, surveillance and military applications, and digital cinema. Three basic feature sets called *profiles* were established to address these application domains: the Baseline, Main, and Extended profiles. The Baseline profile was designed to minimize complexity and provide high robustness and flexibility for use over a broad range of network environments and conditions; the Main profile was designed with an emphasis on compression *coding efficiency* capability; and the Extended profile was designed to combine the robustness of the Basline profile with a higher degree of coding efficiency and greater network robustness and to add enhanced modes useful for special "trick uses" for such applications as flexible video streaming.

1.2. The FRExt Amendment

While having a broad range of applications, the initial H.264/AVC standard (as it was completed in May of 2003), was primarily focused on "entertainment-quality" video, based on 8-bits/sample, and 4:2:0 chroma sampling. Given its time constraints, it did not include support for use in the most demanding professional environments, and the design had not been focused on the highest video resolutions. For applications such as content-contribution, content-distribution, and studio editing and post-processing, it may be necessary to

- Use more than 8 bits per sample of source video accuracy
- Use higher resolution for color representation than what is typical in consumer applications (i.e., to use 4:2:2 or 4:4:4 sampling as opposed to 4:2:0 chroma sampling format)
- Perform source editing functions such as alpha blending (a process for blending of multiple video scenes, best known for use in weather reporting where it is used to super-impose video of a newscaster over video of a map or weather-radar scene)

- Use very high bit rates
- Use very high resolution
- Achieve very high fidelity – even representing some parts of the video losslessly
- Avoid color-space transformation rounding error
- Use RGB color representation

To address the needs of these most-demanding applications, a continuation of the joint project was launched to add new extensions to the capabilities of the original standard. This effort took about one year to complete – starting with a first draft in May of 2003, the final specification was completed in July of 2004, and the editing period was completed in August of 2004. These extensions, originally known as the "professional" extensions, were eventually renamed as the "fidelity range extensions" (FRExt) to better indicate the spirit of the extensions.

In the process of designing the FRExt amendment, the JVT was able to go back and re-examine several prior technical proposals that had not been included in the initial standard due to scheduling constraints, uncertainty about benefits, or the original scope of intended applications. With the additional time afforded by the extension project, it was possible to include some of those features in the new extensions. Specifically, these included:

- Supporting an adaptive block-size for the residual spatial frequency transform,
- Supporting encoder-specified perceptual-based quantization scaling matrices, and
- Supporting efficient lossless representation of specific regions in video content.

The FRExt project produced a suite of four new profiles collectively called the *High* profiles:

- The High profile (HP), supporting 8-bit video with 4:2:0 sampling, addressing high-end consumer use and other applications using high-resolution video without a need for extended chroma formats or extended sample accuracy
- The High 10 profile (Hi10P), supporting 4:2:0 video with up to 10 bits of representation accuracy per sample
- The High 4:2:2 profile (H422P), supporting up to 4:2:2 chroma sampling and up to 10 bits per sample, and
- The High 4:4:4 profile (H444P), supporting up to 4:4:4 chroma sampling,up to 12 bits per sample, and additionally supporting efficient lossless region coding and an integer residual color transform for coding RGB video while avoiding color-space transformation error

All of these profiles support all features of the prior Main profile, and additionally support an adaptive transform block-size and perceptual quantization scaling matrices.

Initial industry feedback has been dramatic in its rapid embrace of FRExt. The High profile appears certain to be incorporated into several important near-term application specifications, particularly including

- The HD-DVD specification of the DVD Forum
- The BD-ROM Video specification of the Blu-ray Disc Association, and
- The DVB (digital video broadcast) standards for European broadcast television

Several other environments may soon embrace it as well (e.g., the Advanced Television Systems Committee (ATSC) in the U.S., and various designs for satellite and cable television). Indeed, it appears that the High profile may rapidly overtake the Main profile in terms of dominant near-term industry implementation interest. This is because the High profile adds more coding efficiency to what was previously defined in the Main profile, without adding a significant amount of implementation complexity.

2. CODING TOOLS

At a basic overview level, the coding structure of this standard is similar to that of all prior major digital video standards (H.261, MPEG-1, MPEG-2 / H.262, H.263 or MPEG-4 part 2). The architecture and the core building blocks of the encoder are shown in Fig. 1 and Fig. 2, indicating that it is also based on motion-compensated DCT-like transform coding. Each picture is compressed by partitioning it as one or more slices; each slice consists of macroblocks, which are blocks of 16x16 luma samples with corresponding chroma samples. However, each macroblock is also divided into sub-macroblock partitions for motion-compensated prediction. The prediction partitions can have seven different sizes – 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4. In past standards, motion compensation used entire macroblocks or, in the case of newer designs, 16x16 or 8x8 partitions, so the larger variety of partition shapes provides enhanced prediction accuracy. The spatial transform for the residual data is then either 8x8 (a size supported only in FRExt) or 4x4. In past major standards, the transform block size has always been 8x8, so the 4x4 block size provides an enhanced specificity in locating residual difference signals. The block size used for the spatial transform is always either the same or smaller than the block size used for prediction. The hierarchy of a video sequence, from sequence, to samples is given by:

sequence (pictures (slices (macroblocks (macroblock partitions (sub-macroblock partitions (blocks (samples)))))).

In addition, there may be additional structures such as packetization schemes, channel codes, etc., which relate to the delivery of the video data, not to mention other data streams such as audio. As the video compression tools primarily work at or below the slice layer, bits associated with the slice layer and below are identified as Video Coding Layer (VCL) and bits associated with higher layers are identified as Network Abstraction Layer (NAL) data. VCL data and the highest levels of NAL data can be sent together as part of one single bit stream or can be sent separately. The NAL is designed to fit a variety of delivery frameworks (e.g., broadcast, wireless, on media). Herein, we only discuss the VCL, which is the heart of the compression capability. While an encoder block diagram is shown in Fig. 1, the decoder conceptually works in reverse, comprising primarily an entropy decoder and the processing elements of the region shaded in Fig. 1.

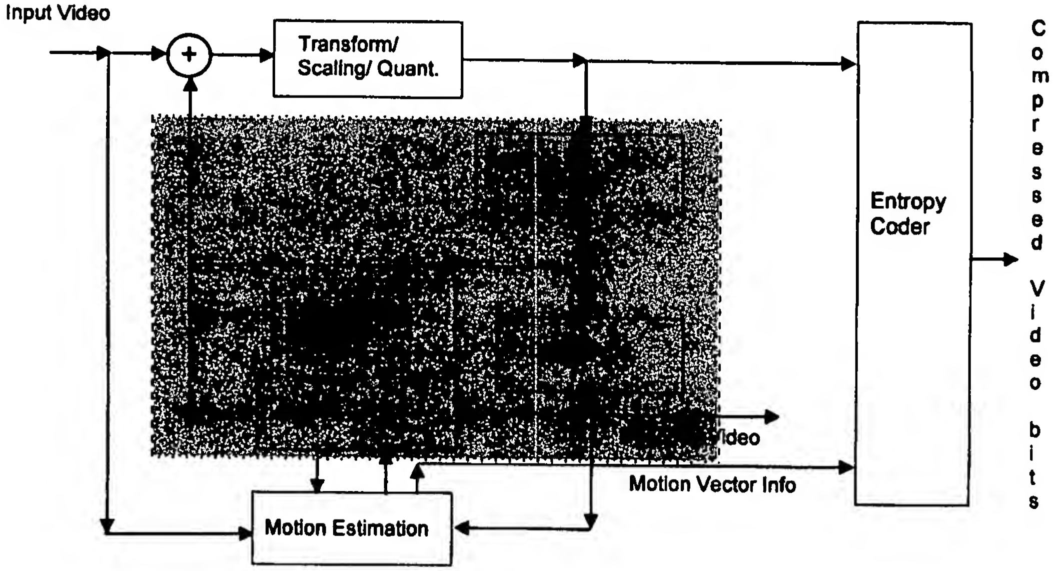

Fig. 1: High-level encoder architecture

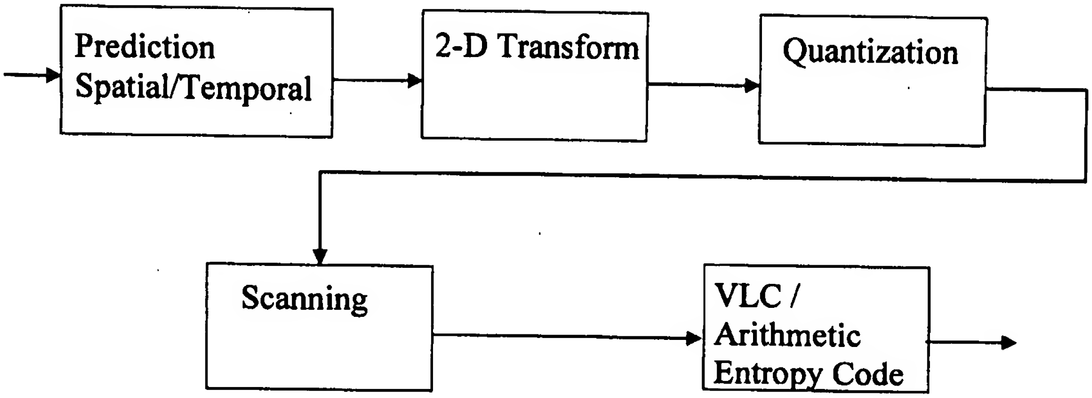

Fig. 2: Higher-level encoder block diagram

In the first version of the standard, only the 4:2:0 chroma format (typically derived by performing an RGB-to-YCbCr color-space transformation and subsampling the chroma components by a factor of 2:1 both horizontally and vertically) and only 8 bit resolution for luma and chroma values was supported. The FRExt amendment extended the standard to 4:2:2 and 4:4:4 chroma formats and higher than 8 bits resolution, with optional support of auxiliary pictures for such purposes as alpha blending composition.

The basic unit of the encoding or decoding process is the macroblock. In 4:2:0 chroma format, each macroblock consists of a 16x16 region of luma samples and two corresponding 8x8 chroma sample arrays. In a macroblock of 4:2:2 chroma format video, the chroma sample arrays are 8x16 in size; and in a macroblock of 4:4:4 chroma format video, they are 16x16 in size.

Slices in a picture are compressed by using the following coding tools:

- "Intra" spatial (block based) prediction
  - Full-macroblock luma or chroma prediction – 4 modes (directions) for prediction
  - 8x8 (FRExt-only) or 4x4 luma prediction – 9 modes (directions) for prediction
- "Inter" temporal prediction – block based motion estimation and compensation
  - Multiple reference pictures
  - Reference B pictures
  - Arbitrary referencing order
  - Variable block sizes for motion compensation
    - Seven block sizes: 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4
  - 1/4 sample luma interpolation (1/4 or 1/8th-sample chroma interpolation)
  - Weighted prediction
  - Frame or Field based motion estimation for interlaced scanned video
- Interlaced coding features
  - Frame-field adaptation
    - Picture Adaptive Frame Field (PicAFF)
    - MacroBlock Adaptive Frame Field (MBAFF)
  - Field scan
- Lossless representation capability
  - Intra PCM raw sample-value macroblocks
  - Entropy-coded transform-bypass lossless macroblocks (FRExt-only)
- 8x8 (FRExt-only) or 4x4 Integer Inverse Transform (conceptually similar to the well-known DCT)
- Residual color transform for efficient RGB coding without conversion loss or bit expansion (FRExt-only)
- Scalar quantization
- Encoder-specified perceptually weighted quantization scaling matrices (FRExt-only)
- Logarithmic control of quantization step size as a function of quantization control parameter

- Deblocking filter (within the motion compensation loop)
- Coefficient scanning
  - Zig-Zag (Frame)
  - Field
- Lossless Entropy coding
  - Universal Variable Length Coding (UVLC) using Exp-Golomb codes
  - Context Adaptive VLC (CAVLC)
  - Context-based Adaptive Binary Arithmetic Coding (CABAC)
- Error Resilience Tools
  - Flexible Macroblock Ordering (FMO)
  - Arbitrary Slice Order (ASO)
  - Redundant Slices
- SP and SI synchronization pictures for streaming and other uses
- Various color spaces supported (YCbCr of various types, YCgCo, RGB, etc. – especially in FRExt)
- 4:2:0, 4:2:2 (FRExt-only), and 4:4:4 (FRExt-only) color formats
- Auxiliary pictures for alpha blending (FRExt-only)

Of course, each slice need not use all of the above coding tools. Depending upon on the subset of coding tools used, a slice can be of I (Intra), P (Predicted), B (Bi-predicted), SP (Switching P) or SI (Switching I) type. B-slices come in two flavors – reference or non-reference. Reference B-slices can be used as reference for temporal prediction. A picture may contain different slice types. In the next section we describe the coding tools used for these different slice types.

This standard is designed to perform well for both progressive-scan and interlaced-scan video. In interlaced-scan video, a frame consists of two fields – each captured at ½ the frame duration apart in time. Because the fields are captured with significant time gap, the spatial correlation among adjacent lines of a frame is reduced in the parts of picture containing moving objects. Therefore, from coding efficiency point of view, a decision needs to be made whether to compress video as one single frame or as two separate fields. H.264/AVC allows that decision to be made either independently for every two-vertical-macroblock pair or independently for each entire frame. When the decisions are made at the macroblock-pair level, this is called MacroBlock Adaptive Frame-Field (MBAFF) coding and when the decisions are made at the frame level then this is called Picture-Adaptive Frame-Field (PicAFF) coding. Notice that in MBAFF, unlike in the MPEG-2 standard, the frame or field decision is made for the vertical macroblock-pair and not for each individual macroblock. This allows retaining a 16x16 size for each macroblock and the same size for all sub-macroblock partitions – regardless of whether the macroblock is processed in frame or field mode and regardless of whether the mode switching is at the picture level or the macroblock-pair level.

2.1. I-slice

In I-slices (and in intra macroblocks of non-I slices) pixel values are first spatially predicted from their neighboring pixel values. After spatial prediction, the residual information is transformed using a 4x4 transform or an 8x8 transform (FRExt-only) and then quantized. In FRExt, the quantization process supports encoder-specified perceptual-based quantization scaling matrices to optimize the quantization process according to the visibility of the specific frequency associated with the transform coefficient. Quantized coefficients of the transform are scanned in one of the two different ways (zig-zag or field scan) and are compressed by entropy coding using one of two methods – CAVLC or CABAC. In PicAFF operation, each field is compressed in a manner analogous to the processing of an entire frame. In MBAFF operation, if a macroblock pair is in field mode then the field neighbors are used for spatial prediction and if a macroblock pair is in frame mode, frame neighbors are used for prediction. The frame or field decision is made before applying rest of the coding tools described below. Temporal prediction is not used in intra macroblocks, but it is for P and B macroblock types, which is the main difference between the slice types. We therefore review the structure of the codec for the I-slice first, and then review the key differences for P and B-slices later.

2.1.1. Spatial Prediction

To exploit spatial correlation among pixels, spatial prediction modes are defined:

- Full-macroblock prediction for 16x16 luma or the corresponding chroma block size, or
- 8x8 luma prediction (FRExt-only), or
- 4x4 luma prediction.

For luma, the full-macroblock prediction is 16x16 in size.

In 16x16 spatial prediction mode, the luma values of an entire 16x16 macroblock are predicted from the pixels around the edges as shown in the Fig. 3. Prediction can be made in one of the four different ways: (i) vertical, (ii) horizontal, (iii) DC and (iv) planar. In the vertical and horizontal predictions the luma values of a macroblock are predicted from the pixels just above or left of the macroblock, respectively. In DC prediction, the luma values of the neighboring pixels are averaged and that average value is used as predictor. In the planar prediction, it is assumed that the macroblock covers diagonally increasing luma values and the predictor is formed based upon the planar equation.

In 4x4 spatial prediction mode, the luma values of 4x4 blocks are predicted from the neighboring pixels above or left of the block and 9 different directional ways of prediction are allowed (see Fig. 3). Each prediction direction specifies a particular set of spatially-dependent linear combinations of previously decoded samples for use as the prediction of each input sample.

8x8 luma intra prediction (available only in FRExt profiles) uses basically the same concept as 4x4 prediction, but with a block size that is 8x8 rather than 4x4 and with low-pass filtering of the predictor to improve prediction performance.

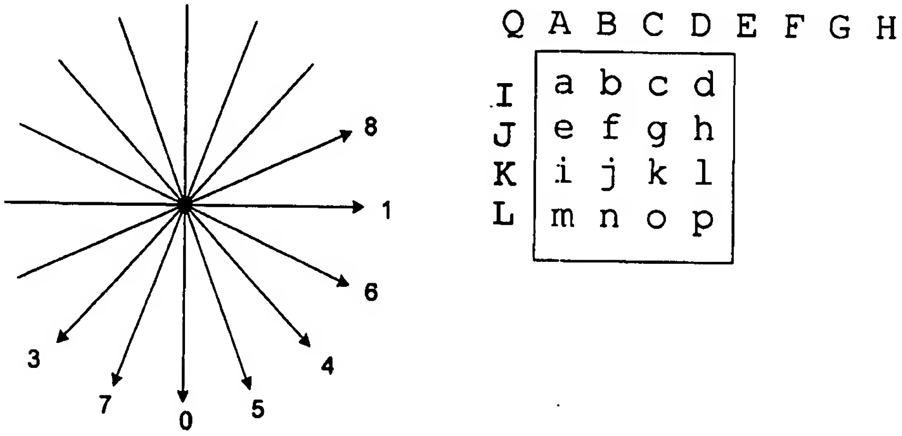

Fig. 3: Spatial prediction of a 4x4 block.

Chroma always operates using full-macroblock prediction. Because of differences in the size of the chroma arrays for the macroblock in different chroma formats (i.e., 8x8 chroma in 4:2:0 macroblocks, 8x16 chroma in 4:2:2 macroblocks, and 16x16 chroma in 4:4:4 macroblocks), chroma prediction is defined for three possible block sizes. In all of these cases the chroma blocks are predicted in a manner similar to the 16x16 luma macroblock prediction: (i) DC, (ii) Horizontal, (iii) Vertical and (iv) planar.

2.1.2. Transform and Quantization

After spatial prediction, a transform is applied to decorrelate the data spatially. There are several unique features about the transform selected for this coding standard. Some of these features are listed below.

(1) It is the first video standard fundamentally based on an *integer* inverse transform design for its main spatial transforms, rather than using a floating point inverse transform. The forward transform that will typically be used for encoding is also an integer transform. MPEG-4 part 2 and JPEG2000 had previously included integer wavelet transforms. But JPEG2000 is an image coding standard without support for interframe prediction, and in MPEG-4, the integer transforms are used only rarely for what is called texture coding (somewhat equivalent to the usual I-frame coding, but not found in most implementations of MPEG-4), and the main transform used for nearly all video data was still specified as a floating point 8x8 IDCT. A significant advantage of this fact is that, with an exact integer inverse transform, there is now no possibility of a mismatch between then encoder and decoder, unlike for MPEG-2 and ordinary MPEG-4 part 2. (The integer transform concept had also been previously applied in H.263 Annex W, but only as an after-the-fact patch to a prior specification in terms of the 8x8 floating point IDCT.)

(2) In fact, the transform is specified so that for 8-bit input video data, it can be easily implemented using only 16-bit arithmetic, rather than the 32-bit or greater precision needed for the transform specified in prior standards.

(3) The transform (at least for the 4x4 block size supported without FRExt) is designed to be so simple that it can be implemented using just a few additions, subtractions, and bit shifts.

(4) A 4x4 transform size is supported, rather than just 8x8. Inconsistencies between neighboring blocks occur at a smaller granularity, and thus tend to be less noticeable. Isolated features can be represented with greater accuracy in spatial location (reducing a phenomenon known as "ringing"). For certain hardware implementations, the small block size may also be particularly convenient.

Thus, while the macroblock size remains at 16x16, these are divided up into 4x4 or 8x8 blocks, and a 4x4 or 8x8 block transform T is applied to every block of pixels. For the 4x4 block size, the transform is specified as follows (other transforms are also defined here for use below):

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

For the 8x8 block size (when selected, and applied only to luma and only with FRExt extensions), a somewhat more complex (but still remarkably simple when compared to an ordinary 8x8 IDCT) transformation matrix is used:

$$T = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & 10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix}$$

The transform T is applied to the luma (16x16) and chroma (8x8, or in FRExt, 8x16 or 16x16) samples for a macroblock by segmenting the full sample block size into smaller blocks for transformation as necessary. In addition, when the 16x16 Intra prediction mode is used with the 4x4 transform, the DC coefficients of the 16 4x4 blocks in that macroblock are further selected and transformed using the Hadamard transform H (note the basic similarity of T and H). Meanwhile, at the general 4x4 level, the corresponding 2x2 chroma samples for 4:2:0 are transformed according to the matrix C

above, a 2x2 Hadamard transform. For 4:2:2 and 4:4:4 chroma formats, the Hadamard block size is increased to reflect the enlarged block shape.

The coefficients after the transformation are quantized using a quantization parameter that can be changed for every macroblock. The parameter can take one of the 52 possible values when video format supports 8 bits per decoded sample. When supporting greater bit depth video content, FRExt expands the number of steps by 6 for each additional bit of decoded sample accuracy. Importantly, the quantization step-sizes are not linearly related to the quantization parameter (as in all prior standards), but vary in such a way that the quantization step size exactly doubles for every 6 increments of the quantization parameter.

A default relationship is specified between the quantization step sizes used for luma and chroma, and the encoder can adjust this relationship at the slice level to balance the desired fidelity of the color components. With the FRExt amendment, the encoder can also balance the Cb and Cr fidelity separately relative to each other.

2.1.3. Perceptual-based quantization scaling matrices

The new FRExt amendment adds support for a feature that had been a mainstay of prior use in MPEG-2 – namely, perceptual-based quantization scaling matrices. The encoder can specify, for each transform block size and separately for intra and inter prediction, a customized scaling factor for use in inverse-quantization scaling by the decoder. This allows tuning of the quantization fidelity according to a model of the sensitivity of the human visual system to different types of error. It typically does not improve *objective* fidelity as measured by mean-squared error (or, equivalently, PSNR), but it does improve *subjective* fidelity, which is really the more important criterion. Default values for the quantization scaling matrices are specified in the standard, and the encoder can choose to instead use customized values by sending a representation of those values at the sequence or picture level.

2.1.4. Scanning

If a macroblock is compressed in the frame mode then the quantized coefficients of the transform are scanned in the zig-zag fashion as shown in Fig. 4a for the 4x4 block size. This scan is designed to order the highest-variance coefficients first and to maximize the number of consecutive zero-valued coefficients appearing in the scan.

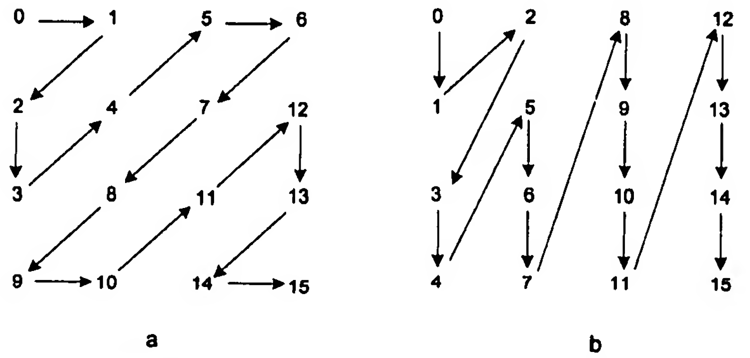

Fig. 4: Coefficient scanning order in (a) Frame and (b) Field modes

If a macroblock is compressed in the field mode then the scanning order of the coefficients is modified to be more efficient for field scanning as shown in Fig. 4b – reflecting the decreased correlation of the source data in the vertical dimension. For other block sizes such as 8x8, the same concepts apply, with similar scans specified for each block size.

2.1.5. Entropy coding

Entropy coding is a lossless coding technique that replaces data elements with coded representations which, in combination with the previously-described predictions and transformations, can result in significantly reduced data size (even though on its own, it can only reduce the data size modestly). Two modes of entropy coding are used in this standard: variable length coding (VLC), a type of Huffman coding, and binary arithmetic coding (BAC). In the current structure, both of these designs are context adaptive (CA), leading to CAVLC and CABAC. Syntax elements at and below the slice layer can be adaptively coded.

Like previous standards, the macroblock is the fundamental unit of the syntax. Data elements to be encoded include:

(a) higher syntax elements for sequence and picture, which are coded using special fixed VLC codes;
(b) slice layer (at and below which layer the entropy codes CAVLC or CABAC are used);
(c) macroblock type (mb_type);
(d) coded block pattern (CBP), which indicates which sets of 4x4 blocks have non-zero elements (thus efficiently indicating which subblocks not to code);
(e) quantization parameter, coded as a delta from the previous macroblock;
(f) reference frame index;
(g) motion vector (MV), sent as a delta from previous MV; and finally
(h) the quantized transform coefficients (from 8x8 or 4x4 transformations, or from secondary Hadamard transformations applied to DC coefficients of lower-level 4x4 transformations).

The final two types of data comprise the bulk of the coded data. At very low bitrates, (g) may be dominant; at all higher rates, (h) is dominant, and it is in encoding the transform coefficients that context adaptivity is primarily employed.

2.1.5.1. CAVLC

The principle idea of VLC is that the data elements to be coded (whether quantized transform coefficients, differential motion vectors, or other syntactic symbols) occur with unequal frequencies; frequently-occurring elements are assigned short codes, while infrequent elements are assigned long codes (thus variable length coding – often called Huffman coding as Huffman codes are the most well-known type of VLCs). For syntax elements other than residual transform coefficients, a fixed VLC is used. Table 1 shows the first nine elements of the Exp-Golomb Code table for given input data elements (here called codeNum). The codeNum is typically an index to the actual data elements (e.g., signed elements such as motion vector differences are mapped by [0, 1, -1, 2, ...] to [0, 1, 2, 3, ...]). These codes have the generic form of [K zeros][1][K-bit DATA], where DATA is a binary representation of an unsigned integer. Such a code is decodable as codeNum = 2^K + int(DATA) –1, where int(DATA) is now the integer corresponding to the binary string. While only the first nine elements are shown, the exp-Golomb code table is conceptually infinite in length. The actual limits on the values of the coded syntax elements are specified by various constraints imposed in the standard.

Table 1: Exponential Golomb Codes (for data elements other than transform coefficients – these codes are actually fixed, and are also called Universal Variable Length Codes (UVLC) [7])

codeNum	code
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001

But for greater efficiency in coding the abundant residual transform coefficients, this standard maintains 11 different sets of codes (4 for the number of coefficients, and 7 for the actual coefficients), which are adapted to the statistics of the current stream or context (thus CAVLC). The latter seven tables are geared towards the lower to higher levels consecutively; coding is typically initialized to lower tables, and incremented up depending on the size of the levels coded. Given the execution efficiency of VLC tables, combined with this limited adaptivity (which thus permits parallelization by macroblocks), this provides a nice tradeoff between speed of execution and performance.

2.1.5.2. CABAC

The use of context-based adaptive binary arithmetic coding (CABAC) has been adopted into the standard as a way of gaining additional performance relative to CAVLC coding, at the cost of additional complexity. The CABAC mode has been shown to increase compression efficiency by roughly 10% relative to the CAVLC mode, although CABAC is significantly more computationally complex. Here the use of arithmetic coding permits non-integer number of bits per symbol, adaptivity allows the coder to adjust to changing symbol statistics, and the context modeling improves prediction performance. CABAC is used for encoding a broader range of syntax elements than CAVLC, starting at the Slice layer (while higher-layer syntax elements are still codec with fixed codes such as Exp-Golomb). In particular, motion vectors as well as residual transform coefficients are codec with CABAC, leading to tighter encoding, whereas with CAVLC only the coefficients are. However, the price to pay is that given the broader applicability of CABAC, combined with its various contexts (e.g., 257 of them in the orginal version of the standard, with a few dozen more added in FRExt), it is basically a serial engine that can be very compute intensive, especially for high pixel and data rates.

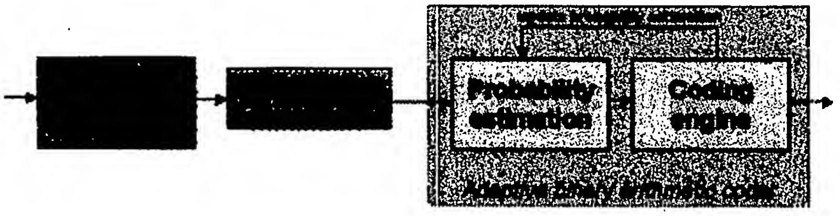

Fig. 5: Generic block diagram of the CABAC entropy coding scheme.

The steps in the CABAC entropy coding scheme are depicted in Fig. 5. Suppose a symbol for an arbitrary syntax element is given. In a first step, a suitable model is chosen according to a set of past observations of relevant syntax elements; this is called *context modeling*. Different models are maintained for each syntax element (e.g., motion vectors and transform coefficients have different models). If a given symbol is non-binary valued, it will be mapped onto a sequence of binary decisions, so-called *bins*, in a second step. The actual *binarization* is done according to a given binary tree – and in this case the CAVLC binary tree is used. Finally, each binary decision is encoded with the *adaptive binary arithmetic coding* (BAC) engine using the *probability estimates*, which have been provided by the context modeling stage. The provided models serve as a probability estimation of the related bins. After encoding of each bin, the related model probability estimate is updated to adjust upward the probability estimate for the binary symbol that was encoded. Hence, the model keeps track of the actual statistics.

Starting with each frame, the probability models associated with all 257 different contexts used in H.254/AVC are initialized with a pre-computed initial distribution. For each symbol encoded, the frequency count of the related binary decision is updated, thus providing a new probability estimate for the next coding decision. However, when the total number of occurrences of a given model exceeds a pre-defined threshold, the frequency counts are scaled down. This periodical rescaling exponentially weighs down past observations and helps to adapt to the non-stationarity of a source. The context modeling used here is innovative, and is described in detail in the standard itself [6]. The arithmetic coding is largely based on the well-known techniques developed in [8].

2.1.6. Lossless macroblock modes

When the fidelity of the coded video is high (i.e., when the quantization step size is very small), it is possible in certain very rare instances of input picture content for the encoding process to actually cause data expansion rather than compression. Furthermore, it is convenient for implementation reasons to have a reasonably-low identifiable limit on the number of bits necessary to process in a decoder in order to decode a single macroblock. To address these issues, the standard includes a "PCM" macroblock mode, in which the values of the samples are sent directly – without prediction, transformation, or quantization. An additional motivation for support of this macroblock mode is to allow regions of the picture to be represented without any loss of fidelity.

However, the PCM mode is clearly not efficient – indeed it is not intended to be efficient – rather, it is intended to be simple and to impose a low bound on the number of bits that can be used to represent a macroblock with sufficient accuracy. If one considers the bits necessary to indicate which mode has been selected for the macroblock, the use of the PCM mode actually results in a minor degree of data expansion. When developing the FRExt amendment, it was decided that a more effective means of lossless coding was desirable for the most demanding applications. FRExt therefore also includes a transform-bypass lossless mode which uses prediction and entropy coding for encoding sample values. When this mode is enabled (which can only be in Hi444P use), the meaning of the smallest selectable value of the quantization parameter is redefined to invoke the lossless coding operation. The new lossless mode of FRExt is a fairly efficient lossless video coder – although not the best method for lossless still-picture (intra) coding, it stands out as extremely efficient when used together with inter-picture prediction (as described in the following sections).

2.2. P-slices

In P-slices (predictively-coded, or "inter" slices), temporal (rather than spatial) prediction is used, by estimating motion among pictures. Innovatively, motion can be estimated at the 16x16 macroblock level or by partitioning the macroblock into smaller regions of luma size 16x8, 8x16, 8x8, 8x4, 4x8, 4x4 (see Fig. 6). A distinction is made between a *macroblock partition*, which corresponds to a luma region of size 16x16, 16x8, 8x16, or 8x8, and *sub-macroblock partition*, which is a region of size 8x8, 8x4, 4x8, or 4x4. When (and only when) the macroblock partition size is 8x8, each macroblock partition can be divided into sub-macroblock partitions. For example, it is possible within a single macroblock to have both 8x8 and 4x8 partitionings, but not 16x8 and 4x8 partitionings. Thus the first row of Fig. 6 shows the allowed macroblock partitions, and the sub-macroblock partitions shown in the second row can be selected independently for each 8x8 region, but only when the macroblock partition size is 8x8 (the last partitioning shown in the first row).

A distinct motion vector can be sent for each sub-macroblock partition. The motion can be estimated from multiple pictures that lie either in the past or in the future in display order. The selection of which reference picture is used is done on the macroblock partition level (so different sub-macroblock partitions within the same macroblock partition must use the same reference picture). The limit on number of pictures used for the motion estimation is specified in the Levels (described below). To estimate the motion, pixel values are first interpolated to achieve quarter-pixel accuracy for luma and up to 1/8th pixel accuracy for chroma. Interpolation of luma is performed in two steps – half-pixel and then quarter-pixel interpolation. Half-pixel values are created by filtering with the kernel [1 -5 20 20 -5 1]/32, horizontally and/or vertically. Quarter-pixel interpolation is performed by averaging two nearby values (horizontally, vertically, or diagonally) of half pixel accuracy. After interpolation, block-based motion compensation is applied. As noted however, a variety of block sizes can be considered, and a motion estimation scheme that optimizes the trade-off between the number of bits necessary to represent the video and the fidelity of the result is desirable.

If a macroblock has motion characteristics that allow its motion to be effectively predicted from the motion of neighboring macroblocks, and it contains no non-zero quantized transform coefficients then it is flagged as skipped. This mode is identified as the Skip mode. Note that, unlike in prior standards, non-zero motion vectors can be inferred when using the Skip mode in P slices.

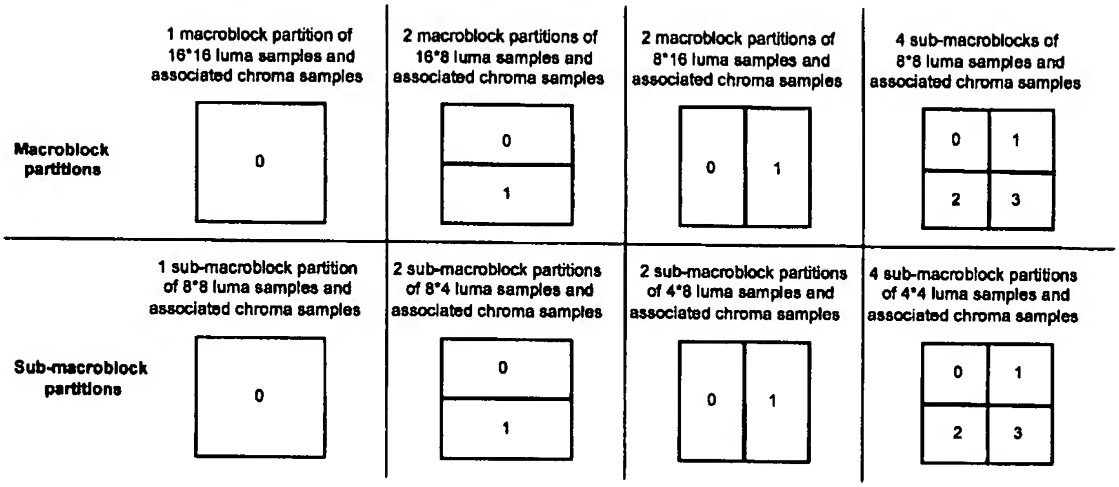
1 macroblock partition of 16*16 luma samples and associated chroma samples
2 macroblock partitions of 16*8 luma samples and associated chroma samples
2 macroblock partitions of 8*16 luma samples and associated chroma samples
4 sub-macroblocks of 8*8 luma samples and associated chroma samples
Macroblock partitions
0
0
1
0
1
0
1
2
3
1 sub-macroblock partition of 8*8 luma samples and associated chroma samples
2 sub-macroblock partitions of 8*4 luma samples and associated chroma samples
2 sub-macroblock partitions of 4*8 luma samples and associated chroma samples
4 sub-macroblock partitions of 4*4 luma samples and associated chroma samples
Sub-macroblock partitions
0
0
1
0
1
0
1
2
3

Fig. 6: Macroblock partitions for motion estimation and compensation

In addition to the use of motion compensation and reference picture selection for prediction of the current picture content, weighted prediction can be used in P slices. When weighted prediction is used, customized weights can be applied as a scaling and offset to the motion-compensated prediction value prior to its use as a predictor for the current picture samples. Weighted prediction can be especially effective for such phenomena as "fade-in" and "fade-out" scenes.

After the temporal prediction, the steps of Transform, Quantization, Scanning, and Entropy coding are conceptually the same as those for I-slices for the coding residual data (the original minus the predicted pixel values). The motion vectors and reference picture indexes representing the estimated motion are also compressed. To compress the motion vectors, a median of the motion vectors from the neighboring three sub-macroblocks – left, above and above right or left – is obtained and the difference from this median vector and the value of the current motion vector is retained and entropy coded. Similarly, the selected reference frame indexes are also entropy coded.

2.3. B-Slices

In B-slices, two motion vectors, representing two estimates of the motion, per sub-macroblock partitions are allowed for temporal prediction. They can be from any reference picture in future or past in display order. Again, a constraint on the number of reference pictures that can be used for motion estimation is specified in the Levels definition. A weighted average of the pixel values in the reference pictures is then used as the predictor for each sample.

B-slices also have a special mode – Direct mode. In this mode the motion vectors for a macroblock are not explicitly sent. The receiver derives the motion vectors by scaling the motion vector of the co-located macroblock in another reference picture. In this case, the reference picture for the current macroblock is the same as that for the co-located macroblock. The motion vector scaling is performed to according to the temporal distances among the current picture, the picture containing the co-located macroblock and the reference picture of that co-located macroblock.

The weighted prediction concept is further extended in the case of B slices. In addition to its use for scaling and offsetting a prediction value, in B slices weighted prediction can enable encoder adjustment of the weighting used in the weighted average between the two predictions that apply to bi-prediction. This can be especially effective for such phenomena as "cross-fades" between different video scenes, as the bi-prediction allows flexible weighted blending of content from such scenes.

Unlike in prior standards, pictures coded using B slices can be used as references for the decoding of subsequent pictures in decoding order (with an arbitrary relationship to such pictures in display order).

2.4. SP and SI Slices

Switching P (SP) and Switching I (SI) slices are close cousins of the usual P and I slices, utilizing either temporal or spatial prediction as before; however, their main virtue is that they can allow reconstruction of specific exact sample values, even when using different reference frames in the prediction process. The main usefulness of this property (which naturally comes at some cost in coding efficiency when compared to the usual P and I slices) is to allow bitstream switching, as well provide additional functionalities such as random access, fast forward, reverse, and stream splicing. These tools are only available in the so-called Extended Profile, and are not allowed in the more common Baseline or Main Profiles. It is unclear at this time what applications will target the Extended Profile, although these tools would be useful for streaming media applications.

2.5. Deblocking Filter

As shown in Fig. 1, H.264/AVC uses an in-loop deblocking filter to reduce the blockiness introduced in a picture. The filtered pictures are used to predict the motion for other pictures. The deblocking filter is an adaptive filter that adjusts its strength depending upon compression mode of a macroblock (Intra or Inter), the quantization parameter, motion vector, frame or field coding decision and the pixel values. For smaller quantization sizes the filter shuts itself off. This filter can also be shut-off explicitly or adjusted in overall strength by an encoder at the slice level.

2.6. Error Resilience Tools

A number of features of the codec design are designed to enable recovery of video fidelity in the presence of network transmission errors or losses. For example, the NAL design, with its highly robust treatment of sequence and picture header content (more properly called sequence and picture parameter sets in the standard), establishes a high degree of robustness. The basic slice structure design adds further robustness, as each slice is designed to completely independent of all other slices of a picture in the basic decoding process (prior to application of the deblocking filter, which can also be made independent by the encoder if desired). No content of any slice of a picture is used for the prediction of syntax elements or sample values used in the decoding process of other slices. Additionally, the encoder can select to indicate that the prediction of intra macroblock sample values in P and B slices will not use spatial neighbors that were not also coded in intra modes – adding further robustness against temporal error propagation. The multiple-reference picture support can also be used by an encoder to enable further resilience against data losses and errors (basically by avoiding the use of any pictures as reference pictures in the prediction process if the fidelity of those pictures may have been adversely affected by transmission errors or losses).

Going beyond these basic feature that are an inherent part of the design, there are essentially four additional tools that are specified in the standard for further protecting the video bitstream from network transmission errors, which may occur for example as a result of congestion overloads on wired networks, or much more frequently due to various channel errors in wireless networks. These tools are: 1) Flexible Macroblock Order (FMO), 2) Arbitrary Slide Order (ASO), and 3) Redundant Slices (RS), and 4) Data Partitioning (DP). FMO and ASO to randomize the data prior to transmission, so that if a segment of data is lost (e.g. a packet, or several continuous packets), the errors are distributed more randomly over the video frames, rather than in a single block of data, making it more likely that relevant neighboring data is available for recovery of lost content. This helps to preserve more local information in all areas, at the cost of some randomly disbributed loss. Redundant Slices offer more protection by reducing the chance of loss via redundancy, a common approach at the level of channel coding. The additional tools of Data Partitioning and SP/SI slices, available in Extended Profile, are also valuable for error resilience/recovery.

2.7. Color Space and Residual Color Transform Support

Like spatial transforms, color transforms to date have generally used floating-point operations and have thus been prone to rounding errors. Typically, video is captured and displayed using the RGB (red, green, and blue) color space, but these components are typically highly correlated. Further, the human visual system seems better matched to luma (brightness) and chroma (hue and saturation) representations, rather than RGB. The usual approach has been to perform a color transformation such as RGB-to-YCbCr before compression, and then code the video in the YCbCr domain, as in:

$$Y = K_R * R + (1 - K_R - K_B) * G + K_B * B; \quad Cb = \frac{(B-Y)}{2(1-K_B)}; \quad Cr = \frac{(R-Y)}{2(1-K_R)};$$

with, e.g., $K_R = 0.2126$, $K_B = 0.0722$.

There are two problems with this approach. The first is that since the samples are actually represented using integers, rounding error is introduced in both the forward and inverse color transformations. The second is that, because the above transformation was not originally designed for digital video compression, it uses a sub-optimal trade-off between the complexity of the transformation (with difficult-to-implement coefficient values such as 0.2126 and 0.0722) and coding efficiency. Focusing on the second problem first, the FRExt amendment adds support for a new color space called YCgCo (where the "Cg" stands for green chroma and the "Co" stands for orange chroma), which is much simpler and typically has equal or better coding efficiency. It uses the following basic equations:

$$Y = \frac{1}{2}\left(G + \frac{(R+B)}{2}\right); \quad Cg = \frac{1}{2}\left(G - \frac{(R+B)}{2}\right); \quad Co = \frac{(R-B)}{2}.$$

While this reduces complexity (and may even improve coding efficiency), it does not, by itself, solve the problem of rounding error. However, rounding errors can be avoided if two additional bits of accuracy are used in the chroma.

FRExt also goes further. Rather that adding two bits of precision to each sample, the FRExt amendment also includes a variant of this scheme which does not require adding precision to the luma samples and which adds only one bit of precision to the chroma samples and does not introduce any conversion rounding error. The equations for the forward transformation in that case are as follows:

$$Co = R - B; \quad t = B + (\,Co >> 1\,); \quad Cg = G - t; \quad Y = t + (\,Cg >> 1\,);$$

where t is an intermediate temporary variable and ">>" denotes an arithmetic right shift operation. For the sake of completeness (so that the reader can check for themselves to see whether the equations are exact integer inverses of each other) we also provide the inverse transformation equations here as follows:

$$t = Y - (\,Cg >> 1\,); \quad G = t + Cg; \quad B = t - (\,Co >> 1\,); \quad R = B + Co.$$

A 1-bit expansion of sample accuracy is still necessary to represent the transformed data in the YCgCo domain in this case, but FRExt has another work-around for this as well. The solution is to retain the use of the RGB domain (in which the sample depth is lower) for the input and output pictures and the stored reference pictures while bringing the above forward and inverse color transformations inside the encoder and decoder for the processing of the residual data only. This technique, called the residual color transform, eliminates color-space conversion error without significantly increasing the overall complexity of the system. Its only drawback is that it can only be applied to 4:4:4 video, as its operation depends on having both luma and chroma samples available for every sample location.

3. SUPPLEMENTAL INFORMATION

In addition to basic coding tools, the H.264/AVC standard enables sending extra supplemental information along with the compressed video data. This ordinarily takes a form called "supplemental enhancement information" (SEI) in the standard. Such data is specified in a backward-compatible way, so that as new types of supplemental information are specified, they can even be used with profiles of the standard that had been previously specified before that definition. The first version of the standard includes the definition of a variety of such SEI data, which we will not specifically review herein. Instead we focus only on what new types of backward-compatible supplemental and auxiliary data are defined in the new FRExt amendment. These new types of data are as follows:

- Auxiliary pictures, which are extra monochrome pictures sent along with the main video stream, and can be used for such purposes as alpha blend compositing (specified as a different category of data than SEI).
- Film grain characteristics SEI, which allow a model of film grain statistics to be sent along with the video data, enabling an analysis-synthesis style of video enhancement wherein a synthesized film grain is generated as a post-process when decoding, rather than burdening the encoder with the representation of exact film grain during the encoding process.
- Deblocking filter display preference SEI, which allows the encoder to indicate cases in which the pictures prior to the application of the deblocking filter process may be perceptually superior to the filtered pictures.
- Stereo video SEI indicators, which allow the encoder to identify the use of the video on stereoscopic displays, with proper identification of which pictures are intended for viewing by each eye.

4. PROFILES AND LEVELS

4.1. The Baseline, Main, and Extended Profiles in the First Version

H.264/AVC contains a rich set of video coding tools. Not all the coding tools are required for all the applications. For example, sophisticated error resilience tools are not important for the networks with very little data corruption or loss. Forcing every decoder to implement all the tools would make a decoder unnecessarily complex for some applications. Therefore, subsets of coding tools are defined; these subsets are called Profiles. A decoder may choose to implement only one subset (Profile) of tools, or choose to implement some or all profiles. The following three profiles were defined in the original standard, and remain unchanged in the latest version:

- Baseline (BP)
- Extended (XP)
- Main (MP)

Table 2 gives a high-level summary of the coding tools included in these profiles. The Baseline profile includes I and P-slices, some enhanced error resilience tools (FMO, ASO, and RS), and CAVLC. It does not contain B, SP and SI-slices, interlace coding tools or CABAC entropy coding. The Extended profile is a super-set of Baseline, and includes B, SP and SI-slices and interlace coding tools, in addition to all of the Baseline Profile's coding tools and further error resilience support in the form of data partitioning (DP). It does not include CABAC. The Main profile includes I, P and B-slices, interlace coding tools, CAVLC and CABAC. It does not include enhanced error resilience tools (FMO, ASO, RS, and DP) or SP & SI-slices.

At this writing, the Baseline Profile appears to be the primary choice for videoconferencing applications. The Main profile, which received a great deal of initial implementation interest for entertainment-quality consumer applications, now seems to be waning in interest due to the new definition of the High profile in FRExt.

Table 2: Profiles in Original H.264/AVC Standard

Coding Tools	Baseline	Main	Extended
I and P Slices	X	X	X
CAVLC	X	X	X
CABAC		X	
B Slices		X	X
Interlaced Coding		X	X
Enh. Error Resil. (FMO, ASO, RS)	X		X
Further Enh. Error Resil (DP)			X
SP and SI Slices			X

4.2. The New High Profiles Defined in the FRExt Amendment

The FRExt amendment defines four new profiles:

- High (HP)
- High 10 (Hi10P)
- High 4:2:2 (Hi422P)
- High 4:4:4 (Hi444P)

All four of these profiles build further upon the design of the prior Main profile, and they all include three enhancements of coding efficiency performance.

- Adaptive macroblock-level switching between 8x8 and 4x4 transform block size
- Encoder-specified perceptual-based quantization scaling matrices
- Encoder-specified separate control of each chroma component quantization parameter

All of these profiles also support monochrome coded video sequences, in addition to typical 4:2:0 video. The difference in capability among these profiles is primarily in terms of supported sample bit depths and chroma formats. However, the High 4:4:4 profile additionally supports the residual color transform and predictive lossless coding features not found in any other profiles. The detailed capabilities of these profiles are shown in Table 3.

Table 3: New Profiles in the H.264/AVC FRExt Amendment

Coding Tools	High	High 10	High 4:2:2	High 4:4:4
Main Profile Tools	X	X	X	X
8x8 vs. 4x4 Transform Adaptivity	X	X	X	X
Quantization Scaling Matrices	X	X	X	X
Separate Cb and Cr QP control	X	X	X	X
Monochrome video format	X	X	X	X
9 and 10 Bit Sample Bit Depth		X	X	X
4:2:2 Chroma Format			X	X
11 and 12 Bit Sample Bit Depth				X
4:4:4 Chroma Format				X
Residual Color Transform				X
Predictive Lossless Coding				X

As can be seen in the table, among these new profiles and the prior Main profile there is a neatly-nested "onion-like" structure of capabilities – with each "higher" profile also supporting all capabilities of the lower ones. Indeed, the standard also requires "higher" profiles to be capable of decoding all bitstreams encoded for the lower nested profiles. At the time of this writing, the High profile seems to be overtaking the Main profile as the primary choice for broadcast and other entertainment-quality applications, and High 4:2:2 is expected to be used frequently in studio environments. The key aspect making the High profile of such intense near-term interest is that it has very little (almost no) added implementation complexity relative to the prior Main profile, while improving compression capability in both subjective and objective terms (with quantization scaling matrices and transform block-size switching, respectively) and increasing encoder control flexibility (with support of separate quantization parameters for the two chroma components).

4.3. Levels

For real-time decoders or decoders with constrained memory size, it is important to specify the processing power and the memory size needed for implementation. Picture size plays the main role in influencing those parameters. As shown in Table 5, H.264/AVC defines 16 different Levels, tied mainly to the picture size. Levels also provide constraints on the number of reference pictures and the maximum compressed bit rate that can be used. The level identified as "1b" was added in the FRExt amendment, primarily to address the expressed needs of some 3G wireless environments. Because the FRExt profiles are specified for more demanding high-fidelity applications, the bit rate capabilities are increased for the FRExt profiles as shown in Table 4, which specifies multipliers for the fourth column of Table 5.

Note: In the standard, levels specify the maximum frame size in terms of only the total number of pixels/frame. Horizontal and Vertical maximum sizes are not specified except for constraints that horizontal and vertical sizes can not be more than Sqrt(maximum frame size * 8). If, at a particular level, the picture size is less than the one in the table, then a larger number of reference pictures (up to 16 frames) can be used for motion estimation and compensation. Similarly, instead of specifying a maximum frame rate at each level, maximum sample (pixel) rate, in terms of macroblocks per second, is specified. If the picture size is smaller than the typical pictures size in Table 5, then the frame rate can be higher than that in Table 5, all the way up to a maximum of 172 frames/sec.

Table 4: Compressed Bit Rate Multipliers for FRExt Profiles (see Table 5 column 4)

FRExt Profile	Bit Rate Multiplier
High	1.25
High 10	3
High 4:2:2	4
High 4:4:4	4

Table 5: Levels in H.264/AVC

Level Number	Typical Picture Size	Typical frame rate	Maximum compressed bit rate (for VCL) in Non-FRExt profiles	Maximum number of reference frames for typical picture size
1	QCIF	15	64 kbps	4
1b	QCIF	15	128 kbps	4
1.1	CIF or QCIF	7.5 (CIF) / 30 (QCIF)	192 kbps	2 (CIF) / 9 (QCIF)
1.2	CIF	15	384 kbps	6
1.3	CIF	30	768 kbps	6
2	CIF	30	2 Mbps	6
2.1	HHR (480i or 576i)	30 / 25	4 Mbps	6
2.2	SD	15	4 Mbps	5
3	SD	30 / 25	10 Mbps	5
3.1	1280x720p	30	14 Mbps	5
3.2	1280x720p	60	20 Mbps	4
4	HD Formats (720p or 1080i)	60p / 30i	20 Mbps	4
4.1	HD Formats (720p or 1080i)	60p / 30i	50 Mbps	4
4.2	1920x1080p	60p	50 Mbps	4
5	2kx1k	72	135 Mbps	5
5.1	2kx1k or 4kx2k	120 / 30	240 Mbps	5

5. SIMULATION RESULTS

5.1. Performance of the First Version of the H.264/AVC

Fig. 7 shows some comparisons of the coding efficiency of MPEG-2, MPEG-4 Part 2 and MPEG-4 Part 10 (H.264/AVC) for the original version of the H.264/AVC specification when tested on a couple of example video sequences. These test results are provided courtesy of the Advanced Technology group of Motorola BCS. In these simulations, no rate control was used and Rate-Distortion (R-D) curves corresponding to encoding with different standards are presented. These are example plots and the results will vary from one encoder to another and from one test video sequence to another. From these plots we see that MPEG-4 Part 2 Advanced Simple Profile (ASP), completed in 1999, provided about 1.5 times coding gain over MPEG-2. And MPEG-4 Part 10 (H.264/AVC), as completed in 2003, provides about 2 times coding gain over MPEG-2 from Rate-Distortion point of view. Note that here the I-frame refresh rate is set at every 15 frames. Since advanced motion estimation is a key strength of H.264/AVC, these results actually *underestimate* its relative merits for broader applications not needing such high intra refresh rates. There are other aspects of the encoding methods used for H.264/AVC in these tests which are also well known to be sub-optimal (particularly its conventional use of non-reference B pictures), so these plots show a conservative estimate.

For the original version of H.264/AVC, most tests seem to show about a 2:1 gain or better in coding efficiency for the new standard. For example, MPEG performed tests that it called "verification tests" in late 2003 [10], and those tests showed similar relative fidelity [11] when using testing methods further described in [12]. The tests done by MPEG measured subjective video quality, unlike what is shown in Fig. 7 (with subjective testing being a better test method, but more difficult to perform rigorously).

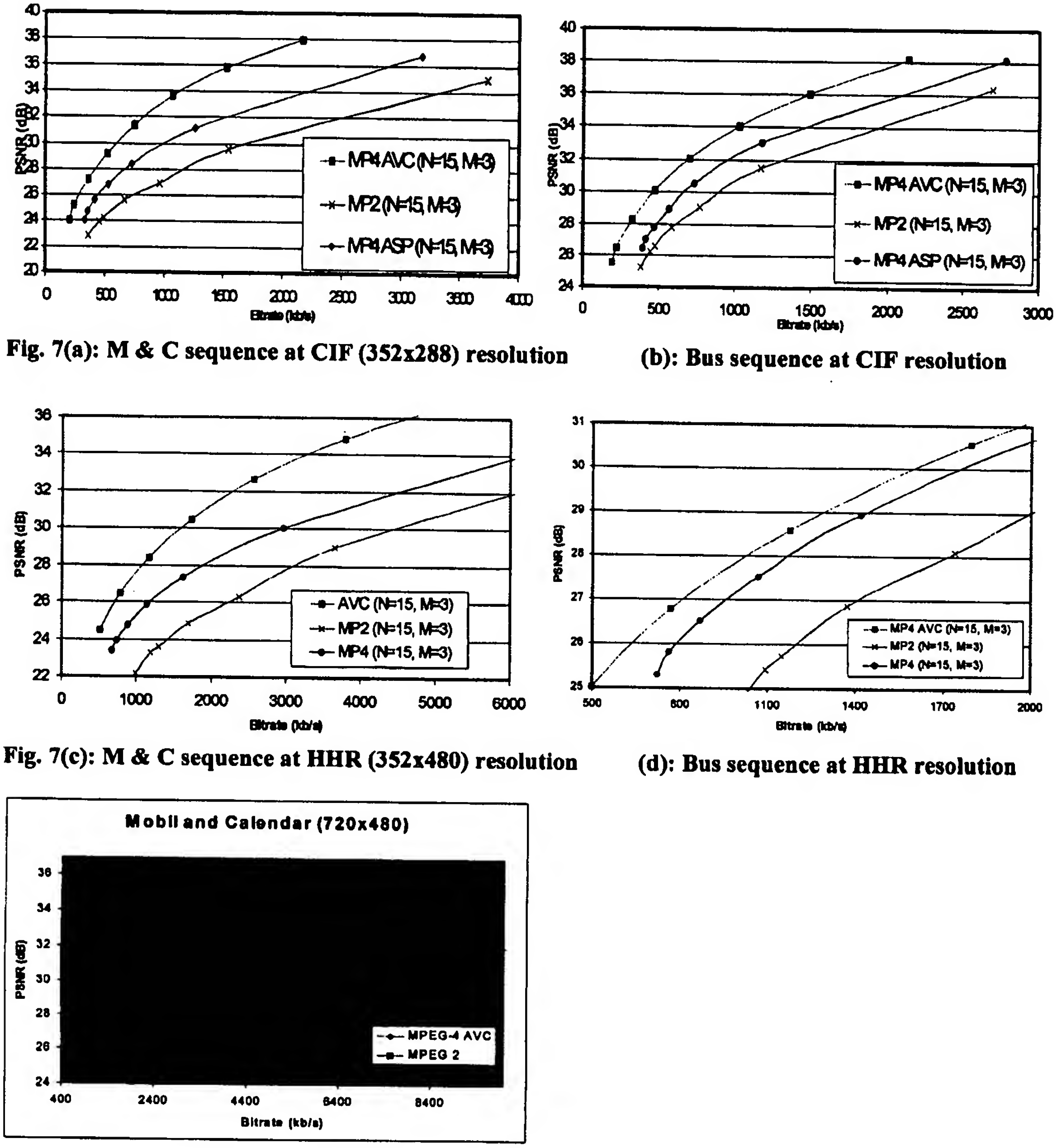

Fig. 7(a): M & C sequence at CIF (352x288) resolution

(b): Bus sequence at CIF resolution

Fig. 7(c): M & C sequence at HHR (352x480) resolution

(d): Bus sequence at HHR resolution

Fig. 7(e): M & C sequence at BT.601 (720x480) resolution

Fig. 7: (a) – (e) Comparison of R-D curves for MPEG-2 (MP2), MPEG-4 ASP (MP4 ASP) and H.264/AVC (MP4 AVC). I frames were inserted every 15 frames (N=15) and two non-reference B frames per reference I or P frame were used (M=3).

5.2. Performance of FRExt High Profile

As FRExt is still rather new, and as some of the benefit of FRExt is perceptual rather than objective, it is somewhat more difficult to measure its capability. One relevant data point is the result of a subjective quality evaluation done by the Blu-ray Disc Association (BDA). The summary results are reproduced in Fig. 8 below (with permission) from [9]. This test, conducted on 24 frame/sec film content with 1920x1080 progressive-scanning, shows the following nominal results (which may or may not be rigorously statistically proven):

- The High profile of FRExt produced nominally *better* video quality than MPEG-2 when using only *one-third* as many bits (*8 Mbps versus 24 Mbps*)
- The High profile of FRExt produced nominally *transparent* (i.e., difficult to distinguish from the original video without compression) video quality at only 16 Mbps.

The quality bar (3.0), considered adequate for use on high-definition packaged media in this organization, was significantly surpassed using only 8 Mbps. Again also, there were well-known sub-optimalities in the H.264/AVC coding method used in these tests. Thus, the bit rate can likely be reduced significantly below 8 Mbps while remaining above the 3.0 quality bar establishing a quality sufficient to call acceptable "HD" in that demanding application.

The result of an example objective (PSNR) comparison test performed by FastVDO is shown in Fig. 9. These objective results confirm the strong performance of the High profile. (Again, sub-optimal uses of B frames and other aspects make the plotted performance conservative for FRExt, thus the remark in the figure about potential future performance.)

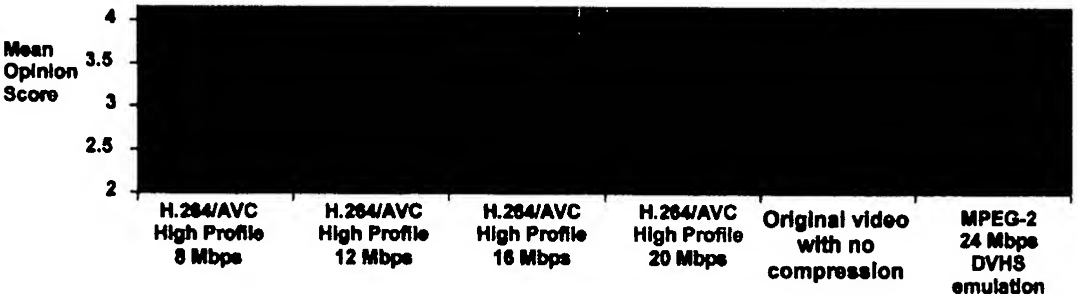

Fig. 8: Perceptual Test of FRExt High Profile Capability by Blu-ray Disc Association [9]

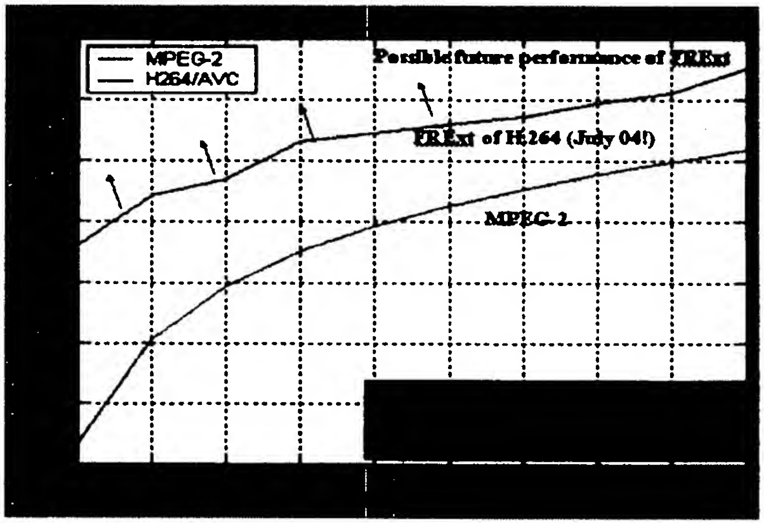

Fig. 9: Objective Performance of FRExt High Profile vs. MPEG-2 on a test 720p clip. These results are consistent with the subjective results in Fig. 8, i.e, 8 Mb/s of FRExt outperforms 20 Mb/s of MPEG-2.

6. CONCLUSIONS

The new video standard known as H.264/AVC presents a rich collection of state-of-the-art video coding capabilities that can provide interoperable video broadcast or communication with degrees of capability that far surpass those of prior standards. With the new FRExt amendment, and especially the new High Profile, H.264/AVC further bolsters its position as the premier design for standardized video compression. We believe these technologies provide a hitherto unavailable set of cost/performance points that will have a powerful impact on both consumer and professional video applications in the years to come.

REFERENCES

[1] ITU-T, "Video codec for audiovisual services at px64 kbits/s," ITU-T Rec. H.261 v1: Nov 1990, v2: Mar. 1993.

[2] ISO/IEC JTC 1, "Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 2: Video," ISO/IEC 11172 (MPEG-1), Nov. 1993.

[3] ITU-T and ISO/IEC JTC 1, "Generic coding of moving pictures and associated audio information – Part 2: Video," ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2), Nov. 1994 (with several subsequent amendments and corrigenda).

[4] ITU-T, "Video coding for low bit rate communication," ITU-T Rec. H.263; v1: Nov. 1995, v2: Jan. 1998, v3: Nov. 2000.

[5] ISO/IEC JTC 1, "Coding of audio-visual objects – Part 2: Visual," ISO/IEC 14496-2 (MPEG-4 Part 2), Jan. 1999 (with several subsequent amendments and corrigenda).

[6] Joint Video Team of ITU-T and ISO/IEC JTC 1, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)," document JVT-G050r1, May 2003; technical corrigendum 1 documents JVT-K050r1 (non-integrated form) and JVT-K051r1 (integrated form), March 2004; and Fidelity Range Extensions documents JVT-L047 (non-integrated form) and JVT-L050 (integrated form), July 2004.

[7] G. Bjøntegaard (Editor), "H.26L Test Model Long Term Number 8", ITU-T/VCEG document. VCEG-N010, San Diego, Sept., 2001.

[8] I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," Comm. of the ACM, vol. 30, no. 6, pp. 520-540, 1987.

[9] T. Wedi and Y. Kashiwagi, "Subjective quality evaluation of H.264/AVC FRExt for HD movie content," Joint Video Team document JVT-L033, July, 2004.

[10] ISO/IEC JTC 1/SC 29/WG 11 (MPEG), "Report of the formal verification tests on AVC/H.264," MPEG document N6231, Dec., 2003 (publicly available at http://www.chiariglione.org/mpeg/quality_tests.htm).

[11] T. Oelbaum, V. Baroncini, T. K. Tan, and C. Fenimore, "Subjective quality assessment of the emerging AVC/H.264 video coding standard," International Brodcasting Conference (IBC), Sept., 2004.

[12] C. Fenimore, V. Baroncini, T. Oelbaum, and T. K. Tan, "Subjective testing methodology in MPEG video verification," SPIE Conference on Applications of Digital Image Processing XXVII, July, 2004.